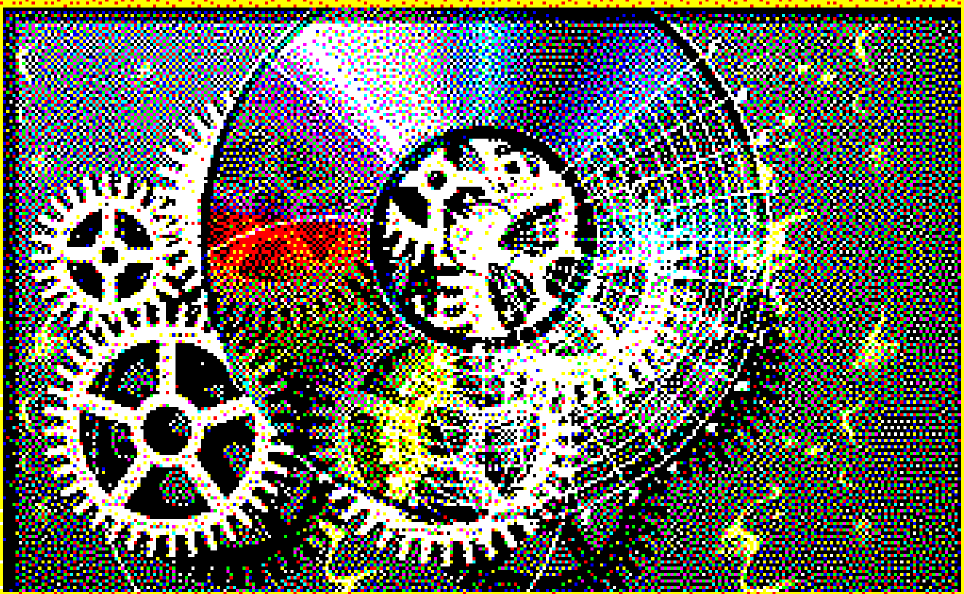
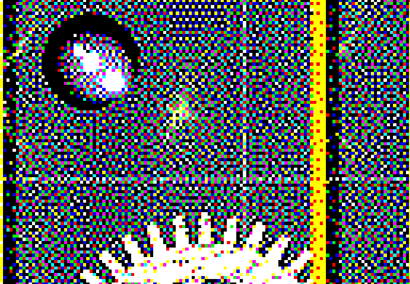
PC-9800シリーズ テクニカルデータブック Windows95編





第1部
Windows95の概要

第1章

Windowsバージョン3.1との比較

1.1　新しく加わった機能

Windows 95では以前のWindowsバージョン3.1のバージョンアップ版という位置付けであるが，フルモデルチェンジと呼べるほどの機能強化，および多数の機能が追加されている．特にWindowsバージョン3.1で不満の多かったプログラムマネージャとファイルマネージャの使い難さは，新しいデスクトップ環境により解消されている．また，MS-DOSから起動しなくてはならなかったことによるメモリやパフォーマンスの制限などの不満も改善された．

それにもかかわらず，Windowsバージョン3.1用に開発された既存の16ビット用アプリケーションやMS-DOSアプリケーションとの高い互換性も保たれている．以前のWindowsからの移行も考慮されており，プログラムマネージャやファイルマネージャも用意されている．

新しく追加および強化されたおもな機能は以下のものである．

- 全面的に改良された新しいユーザーインターフェィス
- 32ビットマルチタスクアプリケーションのサポート
- MS-DOSアプリケーションの互換性の向上
- 安全性の高い，独立したアプリケーションの実行
- 最大255文字まで使用できるファイル名
- プラグ&プレイ
- 32ビット高速デジタルビデオ再生機能の標準装備

・印刷機能の改良
・32ビットネットワークと通信機能の標準装備
・32ビットファイルシステムの導入によるディスクアクセスの向上

第2章

アーキテクチャ

2.1　プリエンプティブなマルチタスク

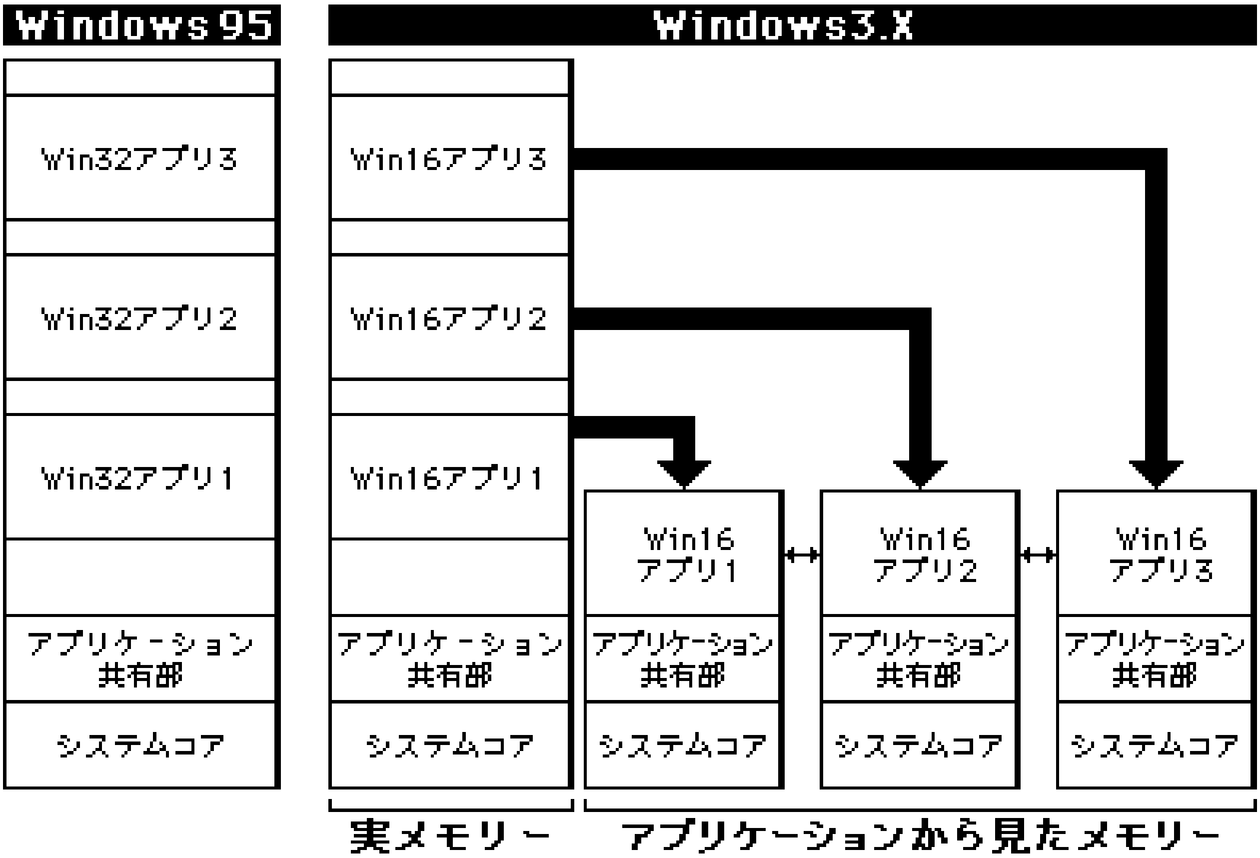
■Windowsバージョン3.1のマルチタスク環境

Windowsバージョン3.1は疑似マルチタスクとでも呼べるようなオペレーティングシステムであった．これはオペレーティングシステムが実行中のアプリケーションを，完全に管理下においていないということを意味している．例えば，CPUの実行時間を占有してしまうようなアプリケーションがあると，ユーザーはそのアプリケーションがCPU時間を解放するまで，他のタスクに切り替えることができなかった．

また，Windowsバージョン3.1用アプリケーションやMS-DOSアプリケーションは16ビットアプリケーションである．CPUは80386以降32ビットアーキテクチャで実行されるが，システムやアプリケーションは実は8086以来の16ビット環境で動作していた．このため，いくつもの制限があった．

Windowsバージョン3.1でもWin32sライブラリを組み込むことにより，32ビットで動作するアプリケーションもある．また，MS-DOSエクステンダ等により32ビットで動作するMS-DOSアプリケーションも存在する．

実メモリとアプリケーション実行時のメモリのイメージ



■Windows 95の32ビットアプリケーション

Windows 95でも従来のWindowsバージョン3.1用アプリケーションはほとんど問題なく動作する．さらにMS-DOS用のアプリケーションも，Windowsバージョン3.1のMS-DOSプロンプトに比べて安定して実行できる．

しかし，Windows 95用アプリケーションは32ビットプロテクトモードのアプリケーションでなければならない．32ビット化されたWindows 95アプリケーションのほとんどは，従来よりも更に高速に動作する．同じマシン上でも処理能力が向上するため，快適性が向上する．

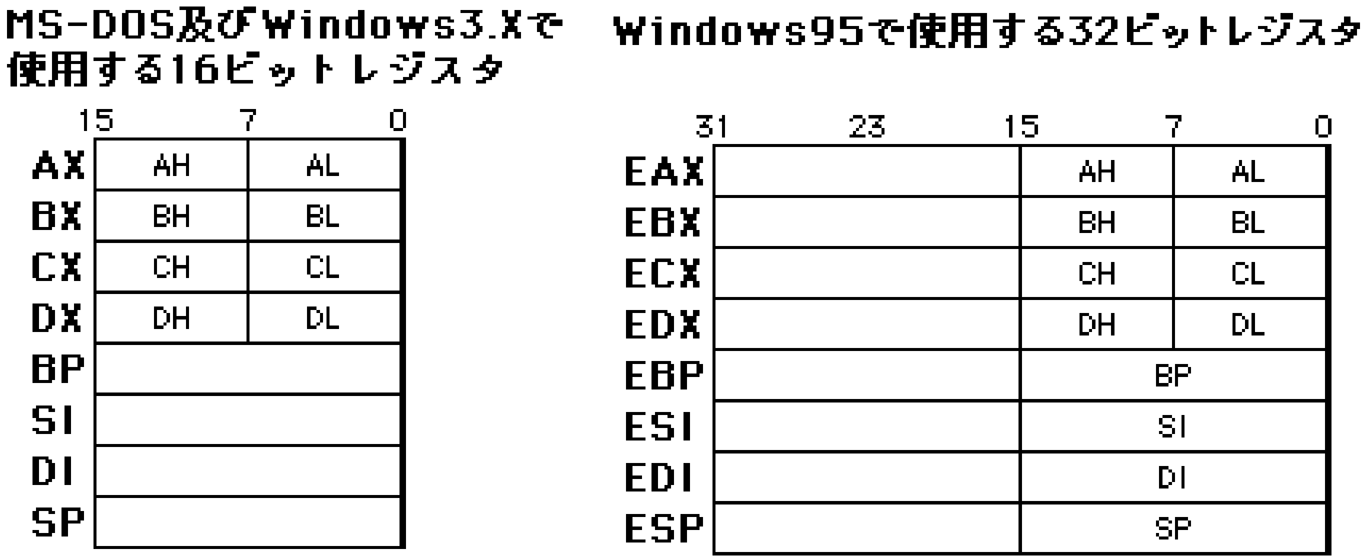
アプリケーションを32ビット化すれば16ビットであったが故の制限が減少する．Windowsバージョン3.1では起動時のMS-DOSのコンベンショナルメモリが少ないと

動作しなかったり，途中でメモリが足りないというエラーになることがあった．また，メモリに余裕があってもシステムリソースが不足し，起動しなかったり，異常動作するアプリケーションがあった．Windows 95のアプリケーションではこれらの問題は解決している．

■16ビットと32ビットアプリケーションの違い

16ビットアプリケーションを実行中は，CPUは16ビットのメモリアドレスをアプリケーション毎に仮想的に割り振らなくてはならなかった．32ビットアプリケーションでは，システムがアプリケーションに単一のメモリ空間を与えるため，APがアドレス空間を意識する必要がなくなった．

また，システム側のモジュールが32ビット化されたため，従来より高速に動作する．



■プリエンプティブなマルチタスク

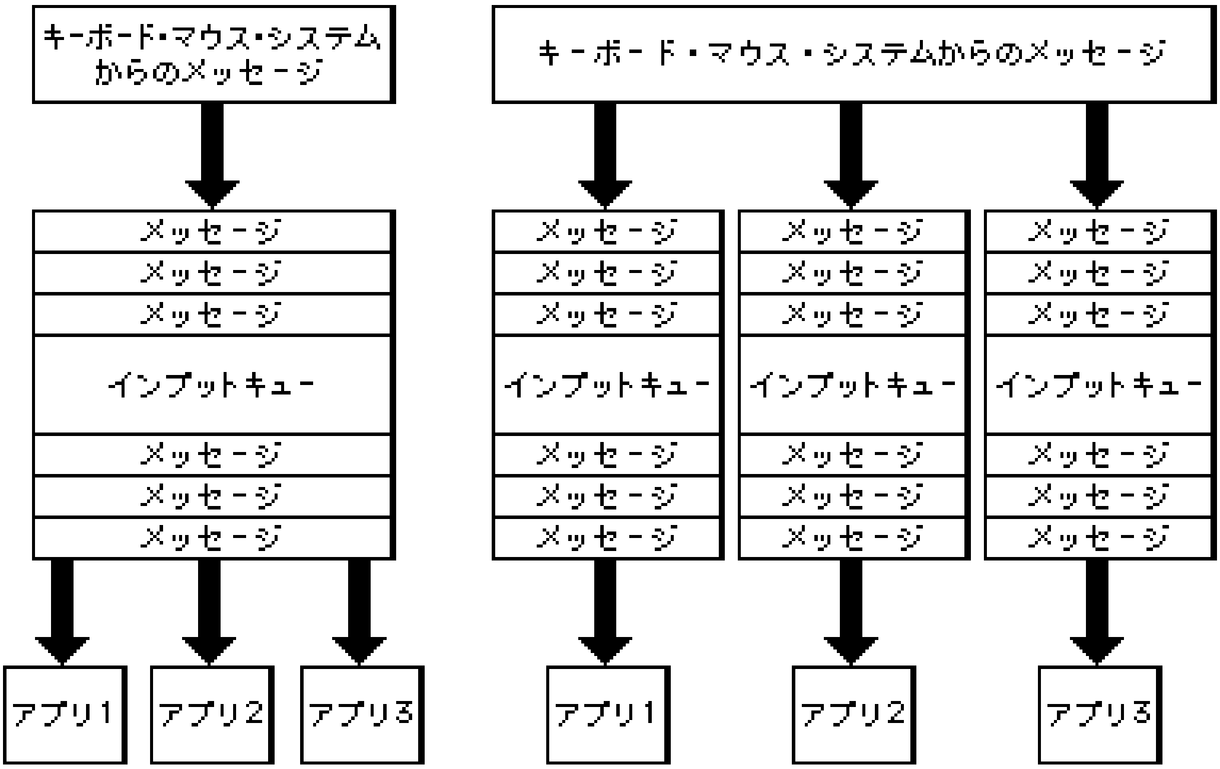
Windowsバージョン3.1の16ビットのメモリ空間を複数用意する疑似マルチタスクに対して，Windows 95はプリエンプティブなマルチタスキングオペレーティン

グシステムである．オペレーティングシステムは実行中のアプリケーションの状態にかかわらず，自由にタスクのスケジューリングができる．

疑似マルチタスクでは1つのアプリケーションがハングアップすると，Windowsの動作まで異常になることがあった．この場合，他のアプリケーションも制御できなくなりデータを失う可能性もあった．また，長いCPU時間を占有してしまうアプリケーションを使用していると，バックグラウンドで動作するプリントマネージャ等のパフォーマンスがかなり低下してしまった．

これらの問題も，プリエンプティブなマルチタスクのスケジューリング機能により解決された．また，１つのアプリケーションが制御不能になったり異常終了しても，あるいはサーバがシステムダウンしてしまっても，他のアプリケーションに被害を及す可能性が低くなり，システムの信頼性が大幅に向上した．

アプリケーション毎に独立したキュー



Windows3.Xでは
メッセージキュー
を共有している

Windows95ではメッセージが
アプリケーションごとに振り分
けられて独立したキューに入る

2.2　ファイルシステムの拡張

Windows 95ではロングファイル名をサポートしたVFATと呼ばれる拡張ファイルシステムが採用されている．WindowsNTのNTFSと似ているが，NTFSはVFATと違いセキュリティが考慮されている．

■ロングファイル名のサポート

MS-DOSの制約であった，ファイル名８文字と拡張子３文字の組み合わせのFATファイルシステムが，Windows 95ではファイル名を255文字まで付けられるようになった．従来は全角文字を使用すると４文字までしかファイル名を付けられず，ファイル名から内容を判断することが難しかったが，このロングファイル名を使用することでファイルの管理や検索が容易になった．ディレクトリ名も同様に255文字まで対応している．

また，MS-DOSで使用できなかった，+,;=[]の６つのキャラクターが新たに使用できるようになった．

ネットワーク上でのロングファイル名の使用であるが，Windows 95同士，あるいはネットワークサーバがロングファイル名をサポートしていれば使用できる．

■既存のアプリケーションとの互換性

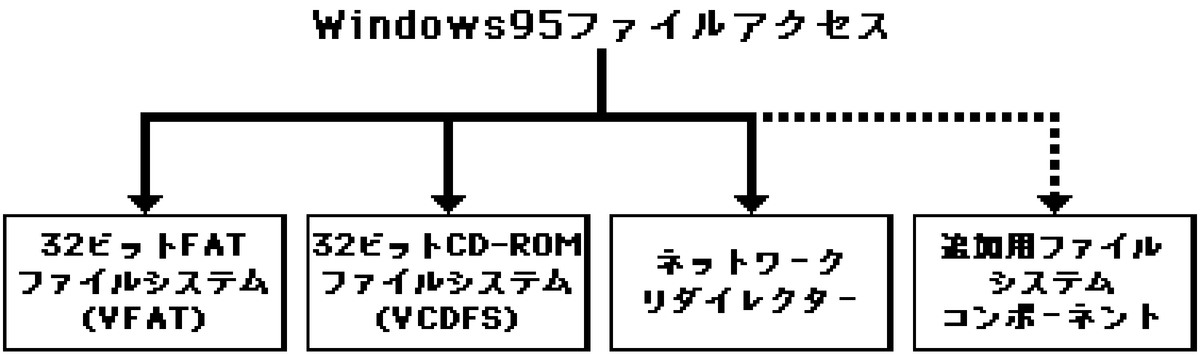
確かにロングファイル名は便利であるが，既存のMS-DOSおよび，Windowsアプリケーションとの互換性が問題になる．この問題は長いファイル名から短い8.3形式のファイルを生成することで解消されている．通常Windows 95上ではロングファイル名を使用し，既存のMS-DOS用およびWindowsバージョン3.1用のアプリケーションで8.3形式を必要とする場合は，Windows 95 のシステムが生成したファイル名を使用する．これは，プロパティーや，MS-DOSプロンプトのDIRコマンドで確認することができる．

また，今までのWindowsバージョン3.1およびMS-DOSとの互換性のため拡張子は必要であるが，Windows 95上のデフォルトでは表示されないようになっている．

■CD-ROMでのロングファイル名の使用

CD-ROMドライブではVFATに似た，CDFSというファイルシステムが採用されている．Windows 95ではCD-ROMドライブが検出されると自動的にCDFSドライバをロードする仕組みになっている．CDFSドライバはプロテクトモード用のMSCDEX.EXEと似たものであるが，CD-ROMドライブが書き込みもサポートしている場合はマルチセッション書き込みができる．

CDFSもVFATと同様にロングファイル名が使用でき，従来からのISO9660形式のCD-ROMも読み込み可能である．Windows 95のMS-DOS互換ボックスでは，ロングファイル名はVFATと同様に8.3形式に変換されたファイルが表示される仕組みになっている．



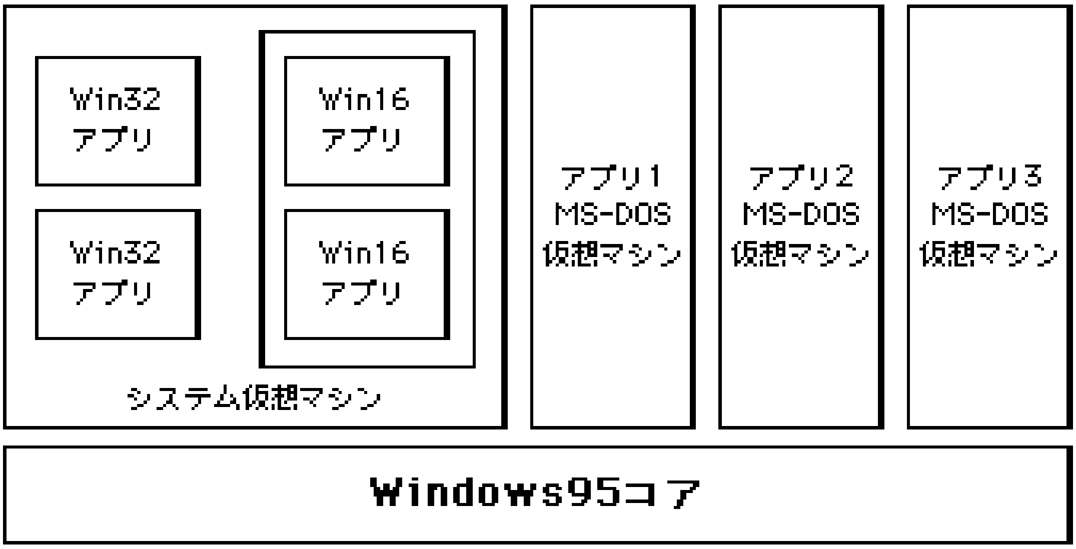
2.3　３つのMS-DOSモード

MS-DOSから起動する必要がなくなったWindows 95だが，既存する多数のMS-DOS用アプリケーションがいまだに存在する．これらのアプリケーションのためにWindows 95では３つのMS-DOSモードが存在する．

複数のMS-DOS互換モードを起動していても，それぞれがローカルなMS-DOS仮想

マシンとして動作するので，以前に比べて安定性が向上している．

　また，MS-DOSアプリケーションそれぞれに対してプロパティーが設定でき，個別のCONFIG.SYSあるいはAUTOEXEC.BATファイルの指定が可能である．



■MS-DOS互換モード（ウインドウ）

　Windows 95のデスクトップ上にウインドウ表示される，MS-DOS互換モードである．このMS-DOS互換モードにはWindowsバージョン3.1から起動するMS-DOSプロンプトと比較して以下のような特徴がある．

- コンベンショナルメモリが広い
- MS-DOSアプリケーションの互換性と安定性の向上
- アプリケーション毎に実行環境を設定するプロパティーのサポート
- ツールバーの利用
- Windowsアプリケーションと同様のMS-DOSアプリケーションの直接終了
- True Typeフォントのサポートと，フォントおよびウインドウサイズの変更

■MS-DOS互換モード（フルスクリーン）

Windows 95から起動するフルスクリーンでのMS-DOS互換モードである．MS-DOSとの互換性はウインドウ表示グラフィックよりさらに高く，グラフィックを使用したゲームやアプリケーションの一部は，このフルスクリーンモードで起動できる．ウインドウ表示のMS-DOS互換モードと比較するとツールバーが表示されないなどの違いはあるが，タスクを切り替えてプロパティーで設定するか，GRPH+RETURNキーでウインドウ表示とフルスクリーン表示を切り替えることができる．

■シングルMS-DOSモード

このモードは前述した２つのモードと違い，Windows 95が切り離されているMS-DOSモードである．ハードウェアを直接制御しているゲームやアプリケーションの場合，MS-DOS互換モードでは正常に動作しない可能性がある．このような場合はシングルMS-DOSモードで実行する必要がある．

プロパティでMS-DOSモード起動用のアイコンを作成し，実行することによりシングルMS-DOSモードになる．Windows 95のコードがメモリから切り離され，フルスクリーンのMS-DOSプロンプトが表示される．EXITコマンドにより再びWindows 95のコードが読み込まれ，デスクトップ画面に戻る．ただし，このモードに移行するにはすべてのプログラムを終了しなければならない．

この他にもシングルMS-DOSモードに移行できる方法がある．コンピューターの起動時の，「Windows95を起動しています」が表示されている時にF8キーを押すことにより，メニューが表示される．このメニューから「Command prompt only」を選択することによりシングルMS-DOSモードになる．この場合，コマンドプロンプトよりコマンドを入力することによりWindows 95が起動する．またWindows 95の終了時に，「MS-DOSモードでコンピューターを再起動する」を選択してもこのモードに移行する．

第3章

ユーザーインターフェース

Windows 95の新しいユーザーインターフェィスが採用され，起動するとデスクトップが表示される．デスクトップ上にはWindowsバージョン3.1と異なり，プログラムマネージャ等は表示されず，アイコンとタスクバーが表示される．
全般的には，多階層のメニューシステムはタブ付きのプロパティー方式に置き換えられ，環境の設定には後戻りのできるウイザードが採用された．また，マウスの右ボタンクリックによるポップアップがシステムとして正式にサポートされた．

■デスクトップ

初期状態ではデスクトップにはマイコンピュータ，ごみ箱と，さらにインストール内容によっていくつかのアイコンと下部にタスクバーが表示されている．
マイコンピュータのアイコンは名前の通り，ユーザー自身のマシンを指す．このアイコンにカーソルをあわせてマウスの左ボタンでダブルクリックすることによりファイルを階層的に探し出し，アプリケーションを実行することができる．以前のディレクトリはフォルダという名前で呼ばれるようになった．
ネットワークに接続されていれば，ネットワークコンピュータのアイコンが表示される．これは，マイコンピュータと同様に，ネットワーク上のコンピュータのリソースを使用することができる．
ごみ箱のアイコンは削除するファイルを一時的に格納するフォルダである．ユーザーが削除したファイルは，このフォルダに保存され，あたかも削除されたかのように見えるが，いつでも必要な時にもとの場所に戻すことができる．ごみ箱を空に

すると完全にハードディスクから削除され，復旧することはできない．ファイルの削除がゴミ箱を経由することで，誤ってファイルを削除する危険が軽減される．

デスクトップ上には，ファイルのアイコンを配置できる．さらに，ファイルそのものではなく，ファイルのショートカットという名前のアイコンを配置することもできる．ショートカットには小さな矢印が表示される．

■タスクバー



Windows 95のデスクトップにはタスクバーが表示されている．タスクバーにはスタートボタンがあり，このボタンをクリックすることで，メニューがポップアップし，ユーザーが行おうとしていることの多くが実行できる．

初期状態のスタートボタンには以下のメニューが表示される．

・プログラム	以前のプログラムマネージャに相当するもの
・最近使ったファイル	以前に使ったファイルが選択可能
・設定	Windows 95の環境設定とメニューのカスタマイズ
・検索	ファイルやフォルダの検索 (複数条件の組み合わせ可能)
・ヘルプ	ヘルプの参照
・ファイル名を指定して実行	アイコンからではなく，ファイル名を指定してアプリケーションを実行
・Windowsの終了	Windowsの終了および再起動

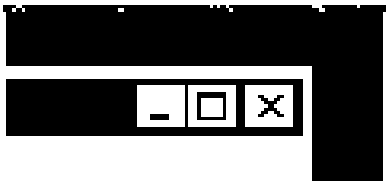
上記のメニュー以外にもインストール方法によってメニューに項目が追加されることがある．ユーザーがカスタマイズすることも可能である．

タスクバーにはスタートボタン以外にも，起動中のアプリケーションや開いているフォルダーが表示され，クリックすることでアプリケーションを切り替えることができる．タスクバーは通常，アプリケーションより前に表示されるので，フォーカスされていないアプリケーションも即座に切り替えることができる．

また、時計が表示されている．さらにアプリケーションやドライバによって，ステータスが表示される．例えば，ボリュームコントロールやバッテリー表示，日本語入力等で使用される．

■ウインドウ

アプリケーションのウインドウには，通常３つのボタンが用意されている．最小化，最大化，終了の３つである．Windowsバージョン3.1のウインドウに加えて，１回のクリックでアプリケーションを終了することができるボタンが追加された．以前のバージョンと同様にウインドウの左上クリックによるウィンドウサイズの変更や終了も可能である．



第4章

プラグ＆プレイ

■プラグ＆プレイのアドバンテージ

パーソナルコンピュータのユーザーは，拡張スロットや拡張バス等に機能拡張のためのデバイスを接続することができる．複数のデバイスを接続すると，I/Oアドレス，IRQ（割り込み要求），メモリ，DMA（ダイレクトメモリアクセス）チャネルのコンフリクト（重複）により正常な動作をしない場合がある．従来はデバイスのマニュアルを参照し，他のデバイスを考慮しながら，設定を手動で行わなくてはならなかった．

Windows 95用のプラグ＆プレイ規格に対応したデバイスは，コンピュータに接続するだけで使用できるようになる．以前のようにディップスイッチ等の設定をせずに，コンピュータが起動するとWindows 95が新しいハードウェアを自動的に認識し，セットアップを行うようになった．マシンにインストールされている機器やドライバを参照しながら最適な設定とドライバの組み込みが行われる．

PCMCIAカードの場合はマシンの電源を入れたまま抜き差しできる．カードを差し込むとドライバが自動的にロードされ，抜くと占有していたメモリが解放される

このプラグ＆プレイ対応の機器には，スロットに差す内蔵カードだけではなく，CD-ROMドライブやCRTモニタ等の外部接続用の機器の規格も存在する．例えばプラグ＆プレイ対応のCRTモニタを使用した場合，Windows 95が自動的に，表示できる周波数や解像度をビデオカードと組み合わせる．ビデオカードで出力できる解像度であっても，CRTモニタが表示できない周波数の場合は，表示されない．

■プラグ＆プレイ対応デバイス

　起動時に自動的に認識されないデバイスを組み込むには，コントロールパネルからハードウェアアイコンを選択することにより，ハードウェアウィザードが起動する．ハードウェアウイザードでは自動的に，あるいは対話的に指定しながら組み込むことができる．

　また，組み込まれているデバイス情報はコントロールパネルのシステムアイコンを選択することにより参照することができる．

　以下は主なハードウェアデバイスの種類別の一覧である．

・キーボード
・サウンド・ビデオ・ゲームのコントローラ
・システムデバイス（主にマザーボード上のデバイス）
・ディスクドライブ
・ディスプレイアダプタ
・ハードディスクコントローラ
・フロッピーディスクコントローラ
・COMポート・LPTポート
・マウス
・モデム
・モニタ
・PCMCIA
・CD-ROMコントローラ
・SCSIコントローラ
・ネットワークアダプタ
・プリンタ
・メモリカード

■プラグ＆プレイ機能に対応していないデバイス

　プラグ&プレイに対応していない従来のデバイスの場合は，手動で設定しなければならない．Windows 95がデバイスを認識すると，レジストリ内にリソース構成の情報が保存されるため，ユーザーは個別に覚えておく必要がなくなった．プラグ&プレイ対応デバイスと非対応のデバイスが混在する場合でも，非対応のデバイスを優先的に割り当てるため設定のコンフリクトが避けられる．今後発売されるデバイスのほとんどはプラグ&プレイに対応するが，新しいデバイスに付属するインストールディスクを使用する必要がある．

　Windows 95にはプラグ&プレイではないがアプリケーションの自動インストール機能も用意されている．CD-ROMドライブにアプリケーションソフトをセットしてドライブアイコンをクリックするだけで，自動的にセットアップが開始される．また，フロッピードライブに未フォーマットのフロッピーディスクをセットし，ドライブアイコンをクリックすると，自動的にフォーマットプログラムが起動し，フォーマットするかどうか確認のメッセージを表示する．

第5章

ネットワーク

Windows 95では標準でいくつかのネットワーク機能が備わっている．ネットワーク機能が装備されたWindows for Workgroupsは日本語版が発売されなかったため，日本ではWindows 95がネットワーク機能を装備した，初めてのWindowsとなる．モデムを介した通信ネットワークとの連携やリモートアクセスもサポートされている．

■ネットワーク標準装備のメリット

Windows95に32ビットネットワーク機能が標準装備されるということは，従来のMS-DOSにネットワークドライバを組み込んでいた時と比較すると，以下のようなメリットがある．

・16ビットネットワークドライバが占有していたMS-DOSのコンベンショナルメモリが解放される
・標準装備のため多くのアプリケーションがネットワークに対応する
・MS-DOSから解放された32ビットネットワークドライバによりネットワーク機能のパフォーマンスが向上する

■複数のネットワークのサポート

Windows 95にはMicrosoft NetworkとNovell Netwareの２つのネットワーク

クライアント機能が組み込まれている．Microsoft NetworkとはMicrosoft LANマネージャやWindows用のネットワークプロトコルであり，Windows 95同士のピアツーピア接続やWindows NT Serverとの接続が可能である．

Novell Netwareとの接続のために，Microsoft Client for Netwareが用意されている．Windows 95ではNetware 3.Xおよび4.Xのサーバにクライアントとして接続できる．もちろんNetwareに接続してあるプリンタを使用可能である．コンベンショナルメモリが広がったMS-DOS互換モードが使用できるメリットも大きい．

Windows 95は複数プロトコルを同時サポートしており，IPX/SPX，TCP/IPやNetBEUIプロトコルが組み込まれている．さらに新たなプロトコルの追加が可能な設計がされている．

■Windows 95のピアツーピアとリモートアクセス

Windows 95マシンにネットワークカードを装着し，イーサーネットケーブルで接続するだけでピアツーピアネットワークを即時構築することができる．ピアツーピアとは，サーバ専用のマシンやＯＳを用意しなくても，それぞれのマシンがファイルやプリンタ等の資源を解放し，サーバの機能を兼ねることのできるネットワークのことである．これによりWindows 95マシン同士でお互いのファイルやプリンタを共有することが可能となる．

モデムを接続することによりリモートアクセスにも対応する．これは1回線のダイアルインゲートウェイとしてWindows 95がサーバの役目をする．このWindows 95が，TCP/IP，IPX/SPX，NetBEUIを使用したネットワークに接続されていれば，移動先からリモートログインできる．もちろん電話回線を通してクライアントとして接続するのもWindows 95でなければならない．

■The Microsoft Network

マイクロソフトはWindows 95にThe Microsoft Network(略してmsnまたはMSネット)とよばれるパソコン通信にアクセスする機能を標準装備した．Windows 95

にモデムと電話回線を用意するだけで，全国各地に存在するアクセスポイントを通じて，msnにアクセスできる.

msnはグラフィカルなパソコン通信で，Windows 95と同じユーザーインターフェィスでサービスを受けることができる．電子メールの送受信もWindows 95内蔵機能をそのまま使用する．インターネットもmsnを通じてアクセスでき，WWW（World Wide Web）にも対応している.

主なサービス内容を下に挙げる．msnは有料サービスであり，サインアップ時にはクレジットカードが必要となる.

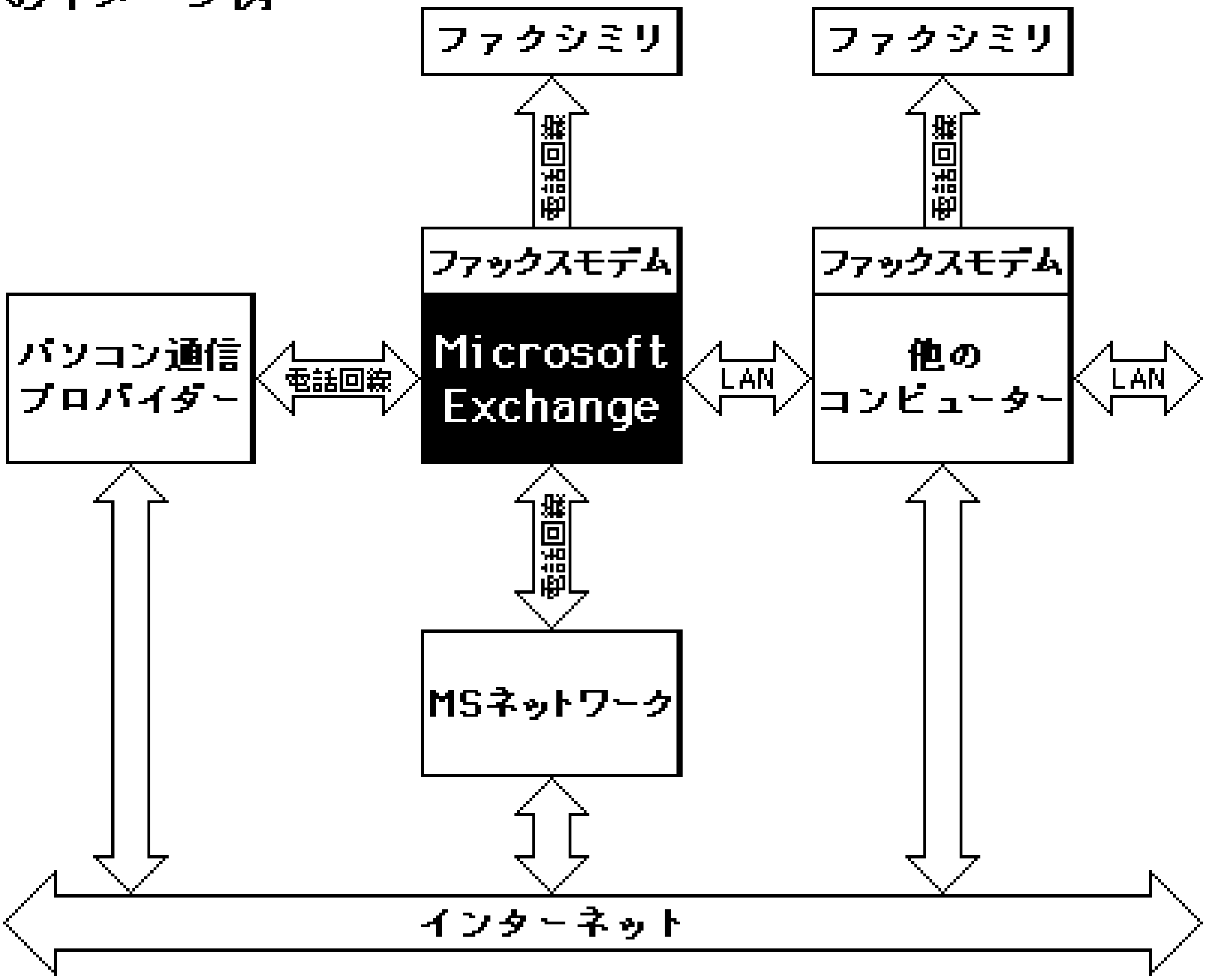
・電子メール
・電子掲示板
・チャット
・ファイルライブラリ
・インターネット接続サービス
・Microsoft 製品情報
・情報サービス
・オンラインショッピング

■Microsoft Exchange

Windows 95にはMicrosoft Exchangeという，メッセージ交換機能が含まれている．具体的にはMicrosoft Mail，The Microsoft Network，Microsoft Faxを通じたメッセージ交換機能を一元化したものを，ひとまとめにしてMicrosoft Exchangeと呼ぶ.

Microsoft Exchangeはメッセージの送受信をまとめたものといえる．例えば，文書を作成した場合，送信先に電子メール（ＬＡN経由または電話回線経由）で送信するか，ＦＡＸで送信するか，文書ファイルそのものを送信するかを，１つのアドレス帳で管理することができる．受信に対しても同様である.

Microsoft Exchange
のイメージ例



第2部
Windows 95対応仕様

第1章

システムデバイス

この章では，PC-9800シリーズの各種システムデバイスの内，Microsoft Windows 95に対応した，主に拡張ボード／カードによって組み込まれるデバイスについて解説する．

本章の内容は，弊社が出版している，「Microsoft Windows 95ハードウェアデザインガイド」との差分情報となっている．本章に記載されていない情報は「Microsoft Windows 95ハードウェアデザインガイド」を参照すること．

なお本章では，フロッピーディスクコントローラ，ATAアダプタ，マウスポート，キーボードポートに関する情報は，PC-9800シリーズ本体側でサポートしているため記述しない．

1.1　一般的なデバイスの要件

PC-9800シリーズにおいてシステムデバイスの拡張には，主にPC98バス(Cバス)，PCIバス，PCMCIAカードの３種類の拡張カード／ボードが利用可能である．

各々の拡張ボード／カードの設計には，次の資料を参照すること．

・PC98バス(Cバス)用ボード：
　-バス信号線/タイミングの詳細は，「HARDWARE編」を参照すること．
　-PC98バスプラグ＆プレイの詳細は，「第3章　PC-98バスプラグ＆プレイ仕様」を参照すること．

・PCIボード：
-PCI-SIGより提供されている仕様書を参照すること.
・PCカード：
-JEIDA Ver4.2/PCMCIA2.1仕様を参照すること.

1.2　ディスプレイアダプタ(グラフィックアクセラレータボード)

PC-9800シリーズ用のディスプレイアダプタ(グラフィックアクセラレータボード)を，PC-9800シリーズ用Windows 95に対応させるためには，以下の機能がサポートされていることを推奨する.

・ディスプレイアダプタは，640×480×8bpp以上の解像度のディスプレイドライバをサポートする必要がある．デスクトップシステムでは，最低256色の表示が可能である必要がある.
・他のデバイスとの競合を避けるために，ディスプレイアダプタで使用する全てのハードウェア資源は，複数の設定が可能である必要がある.
・ディスプレイアダプタ，およびディスプレイアダプタのVDD(Virtual Display Driver)は，VESA Display Data Chanel規格に記述されている，DDC1ホストを満たすことを推奨する.
・ディスプレイアダプタの出力する信号レベルは，0.66V-ppであることを推奨する.

1.3　SCSIホストアダプタ

PC-9800シリーズ用のSCSIホストアダプタを，PC-9800シリーズ用Windows 95に対応させるためには，以下の機能がサポートされていることを推奨する.

・プラグ&プレイ機能を有効にするために，少なくとも4つのI/Oアドレス，7つのIRQ信号を選択できるようにする必要がある．
　また，DMAチャネルを使用する場合，少なくとも2つのDMAチャネルを選択できるようにする必要がある．
・自動的にSCSI IDを割り当てるために，SCAMレベル1プロトコルをサポートすることを推奨する．
・SCSIホストアダプタの中で，内蔵SCSIコネクタと外付けSCSIコネクタを装備するものは，外付け側に自動ターミネーションを使用することを推奨する．
・外付けSCSI機器を接続するためのコネクタは，SCSI-2規格で規定されている形状，または50ピン高密度ベローズコネクタであることを推奨する．
　また，全ての外付けSCSIコネクタは，SPI Annex Fで規定されているSCSIアイコンを表示することを推奨する．

1.4　パラレルポート

　PC-9800シリーズでは，基本的にパラレルポートを標準で1ポートサポートしているが，オプションボードでパラレルポートを拡張することが可能である．
　ここでは，パラレルポート用オプションボードを，PC-9800シリーズ用Windows 95に対応させるために推奨される機能について説明する．

・IEEE P1284規格に記述されているコンパチブルモードおよび，ニブルモードをサポートすることを推奨する．
・ECPモードをサポートすることを推奨する．
・プラグ&プレイ機能を有効にするために，少なくとも2つのI/Oアドレス，7つのIRQ信号を選択できるようにする必要がある．
　また，DMAチャネルを使用する場合，少なくとも2つのDMAチャネルを選択できるようにする必要がある．

1.5　シリアルポート

PC-9800シリーズでは，基本的にシリアルポートを標準で1ポートまたは，2ポートサポートしているが，オプションボードによりシリアルポートを増設することが可能である．

ここでは，シリアルポート用オプションボードを，PC-9800シリーズ用Windows 95に対応させるために推奨される機能について説明する．

- 115.2Kbpsまでの転送速度をサポートしていることを推奨する．
  なお，UARTにはFIFOが付いたものを推奨する．
- プラグ＆プレイ機能を有効にするために，少なくとも4つのI/Oアドレス，7つのIRQ信号を選択できるようにする必要がある．
  また，DMAチャネルを使用する場合，少なくとも2つのDMAチャネルを選択できるようにする必要がある．

1.6　ネットワークアダプタ

PC-9800シリーズ用のネットワークアダプタを，PC-9800シリーズ用Windows 95に対応させるためには，以下の機能がサポートされていることを推奨する．

- NDIS 3.1ネットワークデバイスドライバをサポートする必要がある．
- プラグ＆プレイ機能を有効にするために，少なくとも4つのI/Oアドレス，7つのIRQ信号を選択できるようにする必要がある．
  また，DMAチャネルを使用する場合，少なくとも2つのDMAチャネルを選択できるようにする必要がある．
- ネットワークケーブルが接続されているかを，自動的に検出できるようにすることを推奨する．
- トランシーバタイプを自動検出できることを推奨する．

第2章

デバイスの設計

この章では，PC-9800シリーズに接続可能な各種デバイスを，PC-9800シリーズ用Windows 95に対応させるために，推奨される仕様について記述する．

本章の内容は，弊社が出版している，「Microsoft Windows 95ハードウェアデザインガイド」との差分情報となっている．本章に記載されていない情報は，「Microsoft Windows 95ハードウェアデザインガイド」を参照すること．

2.1　一般的なデバイスの要件

PC-9800シリーズ用Windows 95に，各種デバイスを対応させるためには，プラグ&プレイの機能に必要な以下の最低限の機能を持つことを推奨する．

・各種デバイスは，システムに識別されるものであること．

・外部コネクタを使ってシステムに接続されるデバイスでは，ホットプラギングが可能であること．
ただし，Windows 95ではデバイスはシステムの電源がONのときに接続されたり外されたりした場合，自動的にシステムに認識されるものと，自動的には認識されないものがある．

2.2　ディスプレイモニタ

PC-9800シリーズ用Windows 95に対応するためには，ディスプレイモニタは，以下の機能をサポートすることを推奨する．

・640×480ドット以上の解像度，およびPC98本体の従来互換規格である，640×400ドットの解像度をサポートしていること．
・VESAのwinemphDPMS(Display Power Management Signaling)規格をサポートしていること．
・VESAのMonitor Timing Standards for 640×480,800×600,1024×768,1280×1024 at 75Hz規格，およびVESAの発表する将来のタイミング規格をサポートすること．
・VESAのDisplay Data Channel規格に記されている，DDC1/2Bディスプレイタイプの規格に従うこと．
・通電状態のPC本体に接続することによって，モニタやディスプレイアダプタにダメージを与えないようにすること．
なお，ディスプレイの過電流保護回路は，過電流が流れた場合にユーザがケースを開けて部品を交換する必要が無いものにすること．

2.3　SCSIデバイス

PC-9800シリーズ用Windows 95に対応するためには，SCSIデバイスは以下の機能をサポートすることを推奨する．

・SCSI機器に自動的にSCSI IDを割り当てるために，SCSI自動設定(SCAM)レベル1プロトコルをサポートすることを推奨する．
・外付けSCSI機器のコネクタは，SCSI-2規格で規定されている形状，または50ピン高密度ベローズコネクタであることを推奨する．

また，全ての外付けSCSI機器のコネクタは，SPI Annex Fで規定されているSCSIアイコンを表示することを推奨する.

2.4　ATAデバイス

PC-9800シリーズ用Windows 95に対応するためには，ATAデバイスは以下の機能をサポートすることを推奨する.

・ATA 2仕様のハードウェア，およびソフトウェアの設計に従うこと.
・ATA CD-ROMのATAPI SFF-8020 Ver1.2，またはWestern Digital社製のATAPI CD-ROM規格Rev A1.7に対応する必要がある.

2.5　マウス

PC-9800シリーズ用Windows 95に対応するためには，マウスは以下の機能をサポートしている必要がある.

・PC-9800シリーズのマウス専用バスに接続できること.
・システムに認識されること.
・通電状態の時にマウスの抜き差しを行った場合，マウスにもシステムにも影響を与えないこと.

2.6　キーボード

PC-9800シリーズ用Windows 95に対応するためには，キーボードは以下の機能をサポートしている必要がある.

・PC-9800シリーズのキーボード専用コネクタに接続できること.

・システムに認識されること.
・通電状態の時にキーボードの抜き差しを行った場合，キーボードにもシステムにも影響を与えないこと.

また，PC-9800シリーズ用キーボードでは，以下の機能を追加することを推奨する.
従来のPC-9800シリーズ用キーボードに，以下の3つのキーを追加することにより，キーボードに拡張機能を持たせることが可能となる.

・アプリケーションキー
・右Windowsキー
・左Windowsキー

それぞれのキーのスキャンコードは次のようになる.

キー	スキャンコード（押した場合）	スキャンコード（放した場合）
左Windowsキー	77h	F7H
右Windowsキー	78h	F8H
アプリケーションキー	79h	F9H

第3章

PC98バスプラグ&プレイ仕様

3.1　概要

Cバス（PC-98バス）は98において最も一般的な拡張方式である．このバスアーキテクチャは複数ボードにメモリアドレス，I/Oアドレス，DMAチャネル，割り込みレベルの割り当てを必要とする．しかし，Cバスインターフェィスはこれらのリソースを割り当てるためのハードウェア，およびソフトウェアを持っていない．よって，Cバスボードの設定は通常，メモリやI/Oのデコード，DMAチャネル，割り込みレベルをDIP SWやジャンパによって切り替える．さらに，これらの変更を反映してシステムコンフィグレーションファイルをアップデートする必要があるかもしれない．ユーザーは通常，それぞれのボードメーカーの発行する説明書を参照し，決定する．一般のユーザーにとって，この設定方法は難しく信頼性がない．

他の標準バス(例えばNESA)は，カードのリソース要求を識別し設定するためのハードウェアおよび，ソフトウェアの仕組みを持っている．これらの仕組みはCバススロットのみを持つPC-98と互換性がない．

本書は各ボード間の調整ができるCバス用プラグ＆プレイというハードウェアおよび，ソフトウェア仕様を提案する．すなわち，プラグ＆プレイソフトウェアはユーザーの介在なしに，プラグ＆プレイボードやシステムの他のデバイス間でシステムリソースを最適に割り当てる．なお，これはCバスの物理的･電気的仕様を変更するものではなく，プロトコルの追加である．

プラグ＆プレイボードのみを使ったシステムでは，完全に自動コンフィグレーションを行うことができる．しかし，システム内にプラグ＆プレイボードと一緒

に，従来のCバスボードが存在する場合もある．このようなシステムでは，BIOSやOSにCバスリソースの管理や調整する機能を追加しなければならない．この場合，ユーザー設定が一部必要である．

本書は識別，リソース使用法，競合検出，コンフィグレーションをサポートする必要があるプラグ&プレイボードの仕組みを定義する．本書はまた，ユーザーの介在なしに新しいカードを自動的に設定するプラグ&プレイソフトウェアについても述べる．

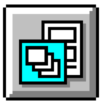
3.2　コンフィグレーション手順

この節では自動コンフィグレーション手順の概要について述べる．ハードウェアおよび，ソフトウェアの詳細は後の節で述べる．

■プラグ＆プレイシステムコンフィグレーション手順

次の図にプラグ＆プレイBIOSを持つシステムでのプラグ&プレイボードのコンフィグレーション処理手順を示す．

▼ *プラグ＆プレイシステムコンフィグレーション手順*



■プラグ＆プレイコンフィグレーション手順

自動コンフィグレーション処理の大まかなステップは次の通りである．

・すべてのプラグ&プレイボードをコンフィグレーションモードにする．
・１つのボードをアイソレートする．
・ハンドルをアサインし，ボードのリソースデータ構造をリードする．

・すべてのボードのリソース要求や機能を取得後，ハンドルを使ってそれぞれのボードにリソースを割り当てる．
・すべてのプラグ＆プレイボードをアクティブにし，コンフィグレーションモードを終了する．

3.3　アイソレーションと認識

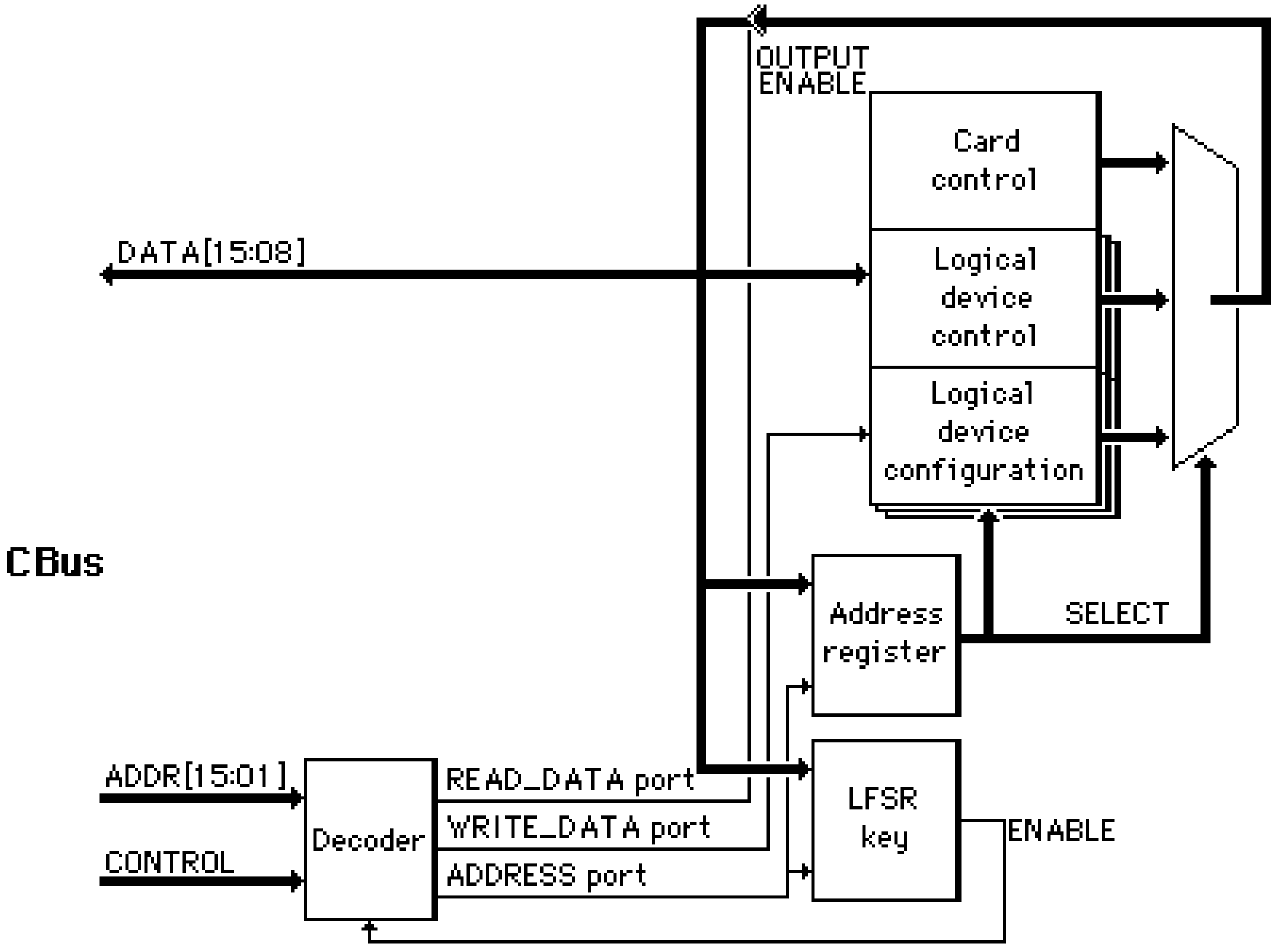
■自動コンフィグレーションポート

それぞれのプラグ＆プレイボードのコンフィグレーションスペースをアクセスするためにソフトウェアは3つの8ビットポートを使用する．これらのポートを次に示す．

自動コンフィグレーションポート

Port Name	Location[1]	Type
ADDRESS	0259H[2]	Write-only
WRITE_DATA	0A59H	Write-only
READ_DATA	0203H-03FFH[3]	Read-only

[1] 16bitデコードする
[2] ISAとはA5を反転
[3] 回路は0003H-03FFHで使用出来るようにする。



●ADDRESS Port

プラグ&プレイレジスタにアクセスするためには，まず要求するレジスタのアドレスをADDRESS Portにライトした後READ_DATA Portからリードまたは，WRITE_DATA Portにライトする．1回のADDRESS Portのアクセスで数回のREAD_DATA Portおよび，WRITE_DATA Portにアクセスをすることができる．

またADDRESS Portはinitiation keyのライトにも使用される．

●WRITE_DATA Port

WRITE_DATA Portはプラグ＆プレイレジスタに情報をライトするために使用する．データはADDRESS Portで最後に指定されたレジスタにライトされる．

●READ_DATA Port

READ_DATA Portはプラグ&プレイレジスタから情報をリードするために使用する．データはADDRESS Portで最後に指定されたレジスタからリードされる．

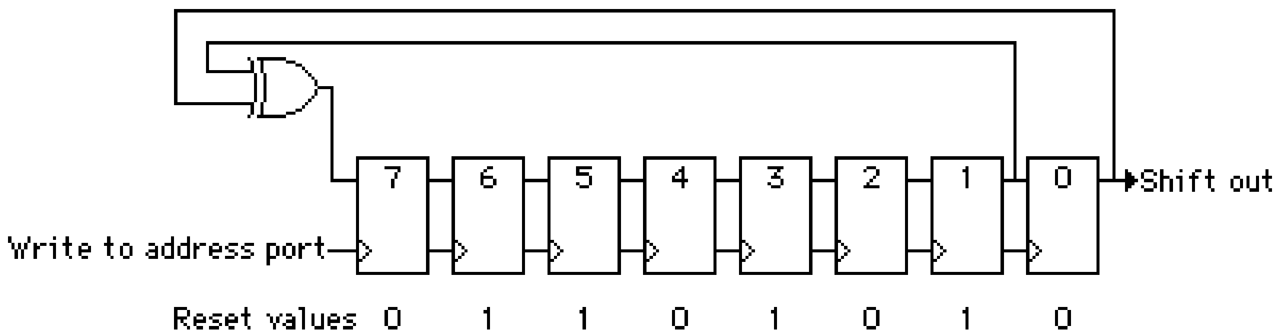
READ_DATA Portのアドレスは，プラグ&プレイレジスタにライトすることで設定する．アイソレーション手順によりREAD_DATA Portが衝突していないか確認する．

■Initiation Key

プラグ&プレイロジックはパワーオンではインアクティブでソフトウェアによりイネーブルにする．

initiation keyはプラグ&プレイロジックをコンフィグレーションモードにする．これはADDRESS Portへの数列のライトによって行われる．ライト数列はボード上のロジックによってデコードされる．もし正しいI/Oライト数列が検出されれば，プラグ&プレイ自動コンフィグレーションポートがイネーブルになる．

initiation key検出のためLFSR(Linear Feedback Shift Register)を使用する．LFSRのハードウェアブロック図を次に示す．



initiation keyの数列(16進数)
6A,B5,DA,ED,F6,FB,7D,BE,
DF,6F,37,1B,0D,86,C3,61,
B0,58,2C,16,8B,45,A2,D1,
E8,74,3A,9D,CE,E7,73,39

ソフトウェアはLFSRシーケンスを作りADDRESS Portに8ビットライトを行う．ハードウェアはライト毎にシフトレジスタの値とライトデータの値を比較する．2つの値が等しくないときはハードウェアはLFSRを初期値6AHにリセットする．ソフトウェアはinitiation keyを送る前にLFSRをリセットするためにADDRESS Portに00Hを2回ライトする．

■アイソレーション手順

自動コンフィグレーションを行うには，それぞれのプラグ＆プレイボードを切り分ける必要がある．これにはCバスの信号を使い，プラグ&プレイのハードウェアとソフトウェアの密着した動作が必要である．

このメカニズムのキーは，それぞれのボードがユニークな番号のシリアルIDを持っていることにある．このシリアルIDは72ビットで，0でない2つの32ビットフィールドと8ビットチェックサムで構成されている．最初の32ビットフィールドはvendor IDである．もう一つの32ビットフィールドは任意の数字で，例えばシリアルナンバやLANアドレスの一部である．シリアルIDはアイソレーションロジックによりビット列でアクセスされ，ボードを区別するために使う．

Check-sum	Serial Number				Vendor ID			
Byte0	Byte3	Byte2	Byte1	Byte0	Byte3	Byte2	Byte1	Byte0
7:0	7:0	7:0	7:0	7:0	7:0	7:0	7:0	7:0

Shift

プラグ&プレイシリアルアイソレーションのシフト順序はbit[0], bit[1]…bit[7]となる.

●ハードウェアプロトコル

アイソレーション処理は，プラグ&プレイソフトウェアによりいつでも呼び出される. initiation keyはすべてのボードをコンフィグレーションモードにする. それぞれのボードのハードウェアはREAD_DATA portへの72組(144回)のI/Oリードアクセスを受ける. ボードの応答はシリアルIDの各ビットによって異なる.

もしその時のシリアルIDが"1"であれば，最初のI/Oリードサイクル時にデータバスに55Hを出力する. もし"0"であればデータバスはハイインピーダンスにする. すべてのハイインピーダンス状態のボードはI/Oサイクル中に他のボードがD[9:8]を"01"にドライブしているかチェックする. 2回目のI/Oリード中は，1回目に55HをドライブしていたボードはAAHをドライブする. すべてのハイインピーダンス状態のボードは他のボードがD[9:8]を"10"にドライブしているかチェックする（リードサイクル中，ハードウェアは8ビットドライブするがチェックは下位2ビットのみ行う）.

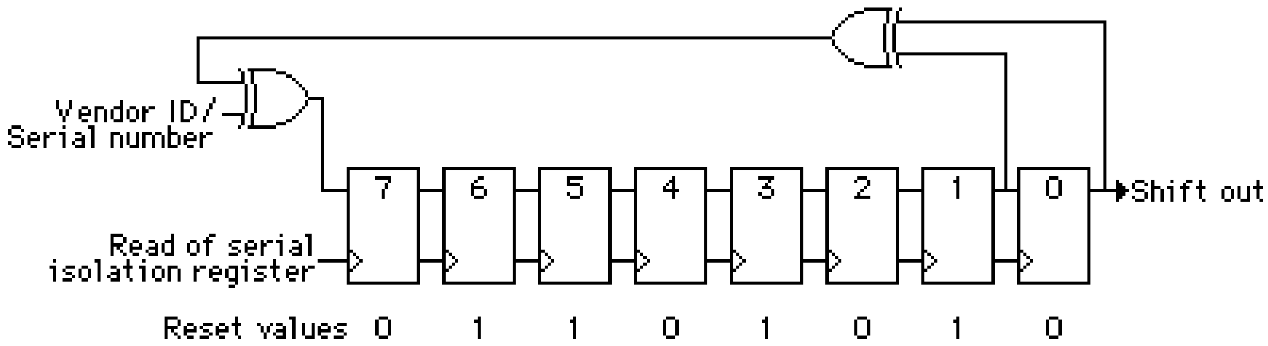
ハイインピーダンスボードは他のボードが正しいデータをドライブしているのを検出した場合，アイソレーション処理を終了する. これらのボードは次サイクルのアイソレーション処理を再び実行する.

バスをドライブしたりハイインピーダンス状態であったが他のボードのドライブを検出できなかった場合は，次のI/Oリードサイクルに備える. ボードはシリアルIDをシフトし，次の応答を決定する.

この手順をシリアルID72ビット分行う.

この処理の終了後1枚のボードが残る．このカードに今後，ボードの選択に使うCSN(Card Select Number)をアサインする．CSNをアサインされたボードは，その後のアイソレーション処理には関与しない．ボードは他のコマンドを受ける前にCSNをアサインされなければならない．

このプロトコルの8ビットチェックサムは，ボード上の不揮発メモリにストアしておいたり，カード上のロジックでリアルタイムに生成することができる．チェックサムの生成にはinitiation keyと同様のLFSRを使用する．チェックサム用LFSRのブロック図を次に示す．



プラグ&プレイボードはアイソレーション中に，IORDY信号をドライブしてはならない．それ以外であればドライブできる．

●ソフトウェアプロトコル

初めにプラグ&プレイソフトウェアは，すべてのプラグ&プレイボードを初期状態にするために後述のリセットコマンドを送る(リセットコマンドを有効にするためにリセットコマンドの発行に先立ち，Wait for Keyコマンドおよびinitiation keyを送る必要がある）．

プラグ&プレイソフトウェアはコンフィグレーションモードにするために，すべてのプラグ&プレイボードにinitiation keyを送る．ソフトウェアはアイソレーション処理を行うために準備する．

プラグ&プレイソフトウェアはREAD_DATA portからの72組のI/Oリードサイク

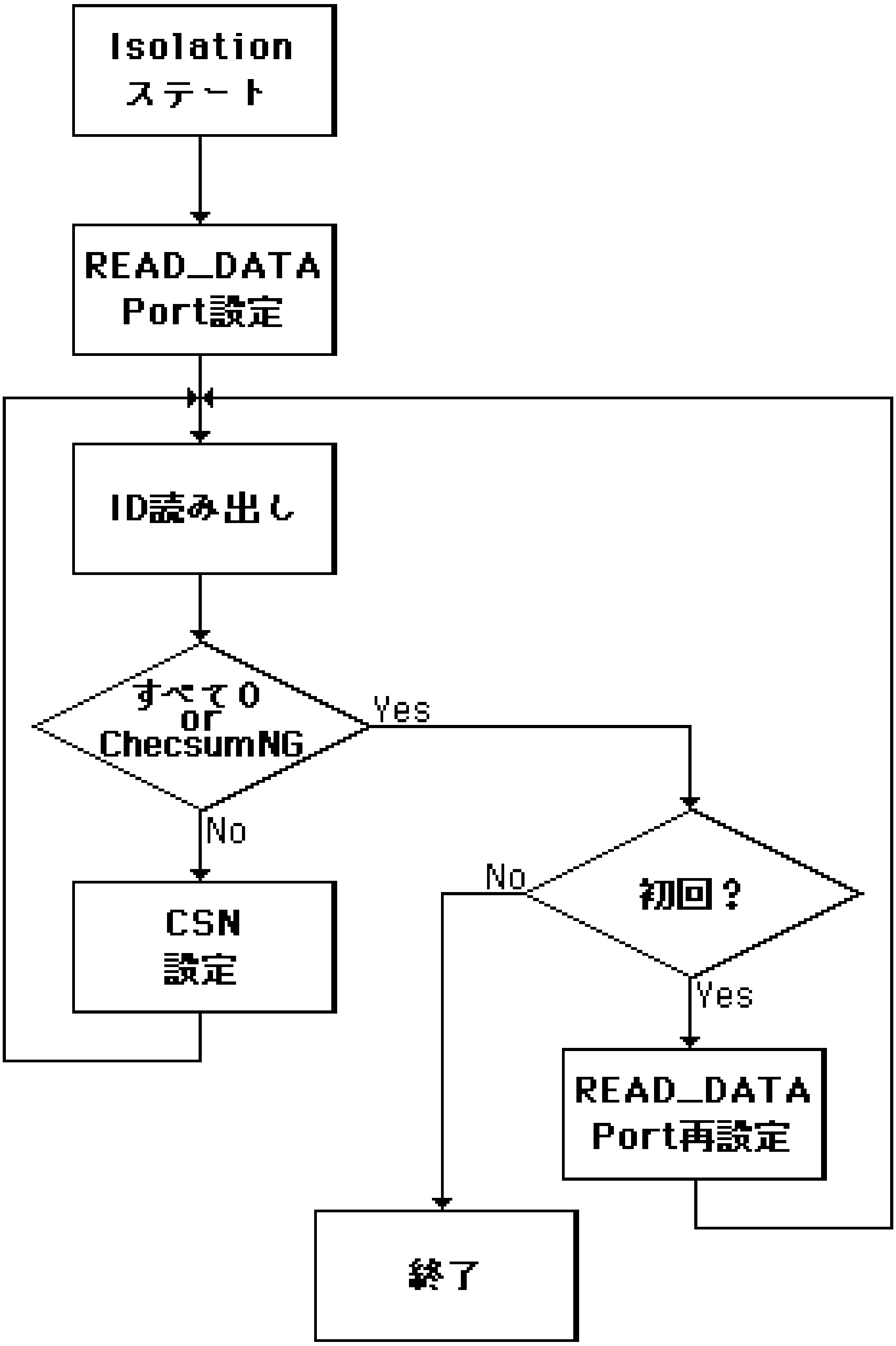
ルを発生する．ソフトウェアはそれぞれのI/Oリードでハードウェアが55H，AAHをドライブしているかチェックする．55H，AAHがリードできた場合，このときのビットは"1"であると仮定する．それ以外は"0"と仮定する．

ソフトウェアは最初の64ビットからチェックサムを生成し，続く8ビットと比較する．

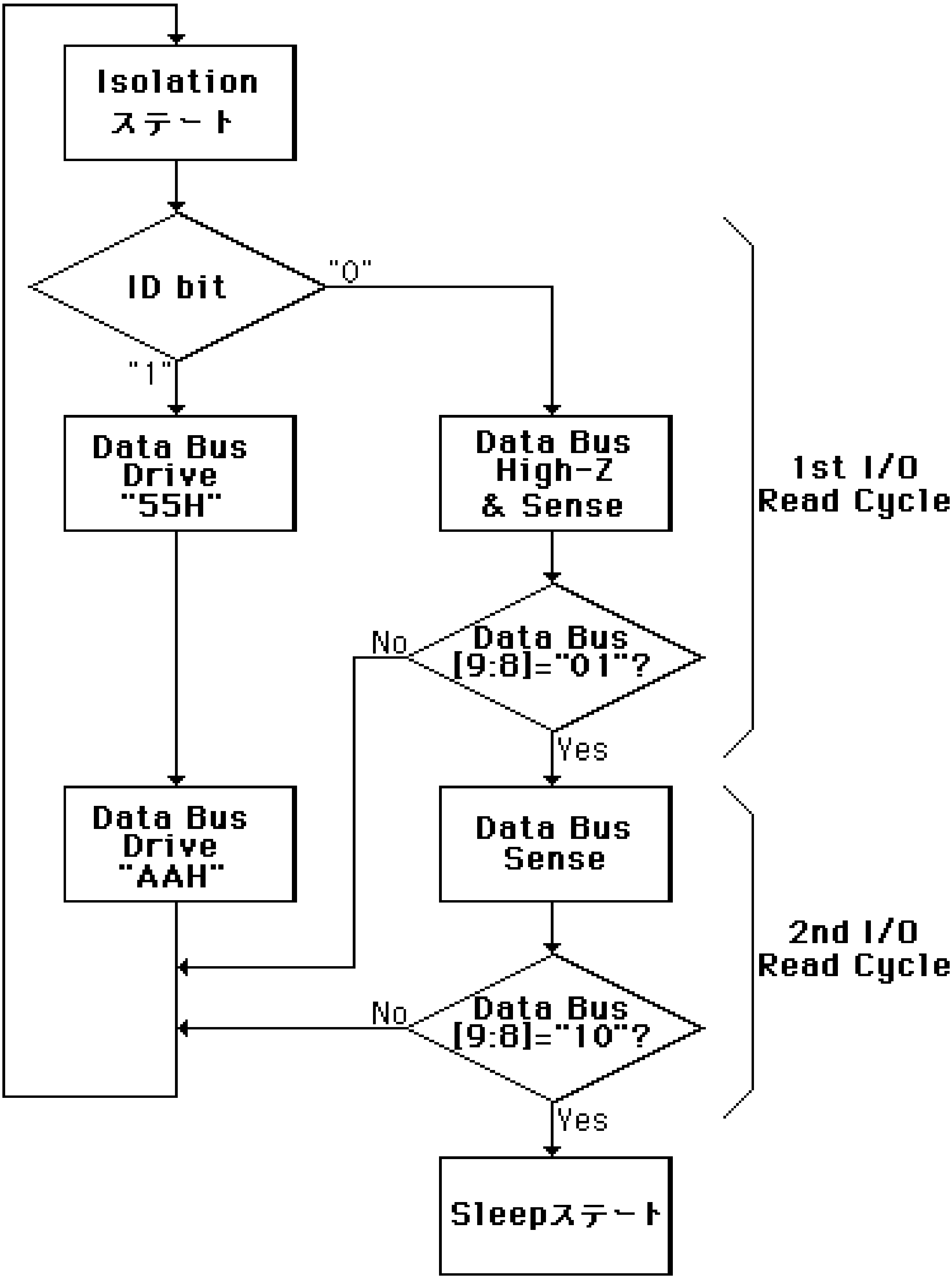
また，ソフトウェアは次のことについて考慮する必要がある．I/Oリードサイクル中，55H，AAHを検出できなかったりチェックサムが合わないことが起こり得る．どちらかが最初のI/Oリードサイクルで起こったらREAD_DATA portが競合していると判断する．競合を検出したら，READ_DATA portを再配置する．この操作を競合しないREAD_DATA portが見つかるまで繰り返す．0203H～03FFHが使用できることになっているが，数回繰り返しても見つからない場合はプラグ&プレイボードは無いものと判断する．

数回シリアルIDのリードを行った後，これらのことが起こったら，もうプラグ&プレイボードは無いと判断する(すなわち前回の操作で最後のボードを見つけた)．これでアイソレーション処理を終了する．

注意：ソフトウェアは最初のアイソレーションリードを始める前に1mSecのウェイトを入れなければならない．また，各アイソレーションシーケンス間に250μSecのウェイトを入れなければならない．



Isolationプロトコル



bit読み出しプロトコル

3.4　プラグ＆プレイデバイスのプログラム

■プラグ＆プレイステート

まずCSN(Card Select Number)について説明する．CSNは1枚(あるステートでは複数枚)のボードを選択するために使用する8ビットレジスタである．CSNはパワーアップ時にすべてのボードが0にセットされる．一度アイソレートされたら，CSNはユニークな番号をアサインされる．ソフトウェアはこの番号で，再びアイソレーション手順を行うことなくコンフィグレーション処理でボードを選択することができる．

プラグ&プレイステートは次の通りである．

- Wait for Key　パワーアップ後すべてのボードは，Wait for Keyコマンドへの応答でこのステートに入る．バスからのinitiation keyを検出するまでこのステートでアクティブになるコマンドはない．プラグ&プレイボードはシステムの通常動作時はデフォルトでWait for Keyステートとなる．
- Sleep　このステートではプラグ＆プレイボードはWake[CSN]コマンドを待つ．このコマンドはライトデータと各ボードのCSNの値によって1枚（以上）のボードをIsolationまたは，Configステートにするために選択する．ボードはWake[CSN]コマンドのbits[7:0]がボードのCSNと一致したときにSleepステートを終了する．Wake[CNS]コマンドのライトデータが00Hの時は，CSNをアサインされていないすべてのボードがIsolationステートになる．00HでないときはWake[CSN]コマンドのパラメータと一致するCSNをアサインされているボードのみがConfigステートになる．
- Isolation　このステートではアイソレートの節で述べたserial isolation registerのリードに応答する．アイソレートされるとユニークなCSNをアサインされる．この番号はWake[CSN]によってボードの選択に使用される．CSNを設定

されるとConfigステートに移行する.
- Config　Configステートにあるボードは，ボードのリソースコンフィグレーション情報のリードや，ボードのリソース選択設定を含むすべてのコンフィグレーションコマンドに応答する.

■プラグ＆プレイレジスタ

プラグ＆プレイボードの標準レジスタ空間は，card control，logical device control，logical device configurationの3つの部分に分かれている．card controlレジスタはそれぞれのボードに1つずつある．card controlレジスタはボード全体の機能のコントロールに使用される．logical control deviceレジスタとlogical device configurationレジスタはlogical device毎にある．logical device controlレジスタはデバイスがバス上で使用できるように機能をコントロールする．logical device configurationレジスタはデバイスのリソース設定に使用される.

■コントロールレジスタ

プラグ&プレイボードはあるバスの状態でプラグ&プレイレジスタにライトされたコマンドに応答する．これらのコマンドの概要は次の通りである.

- RESET　これはバスのリセットである．プラグ&プレイボードがリセット信号を検出したらWait for Keyステートに移行する．すべてのボードのCSNは00Hにリセットされ，logical deviceのコンフィグレーションレジスタはデフォルト値に戻る．すべての非ブートlogical deviceはインアクティブになる．ブートデバイスはアクティブになる.
  ソフトウェアはRESET後，自動コンフィグレーションポートをアクセスする前に1mSecのウェイトを入れなければならない.
- config control register　config control registerは該当するビットに

"1"をライトすることでアクティブになる，3つの独立したコマンドからなっている．これらのビットはコマンド実行後にハードウェアにより自動的に"0"にリセットされる．

・Resetコマンド　Resetコマンドはconfig control registerに01Hをライトすることによってプラグ&プレイボードに送られる．すべてのプラグ&プレイボードはWait for Keyを除くステートでこのコマンドに応答する．このコマンドはすべてのlogical deviceをリセットする．これはコンフィグレーションレジスタの内容をデフォルトに戻す．すべてのlogical deviceのコンフィグレーションレジスタもデフォルト値に戻る．Read Data PortやCSN，プラグ&プレイステートは保持される．
ソフトウェアはResetコマンドを発行した後，自動コンフィグレーションポートをアクセスする前に1mSecのウェイトを入れなければならない．

・Wait for Keyコマンド　Wait for Keyコマンドはconfiguration control registerに02Hをライトすることによりプラグ&プレイボードに送られる．すべてのプラグ&プレイボードは，どのステートでもこのコマンドに応答する．このコマンドはすべてのプラグ&プレイボードをWait for Keyステートにする．CSNは保持されlogical deviceの設定は変わらない．

・Reset CSNコマンド　Reset CSNコマンドはconfiguration control registerに04Hをライトすることによりプラグ&プレイボードに送られる．Wait for Keyを除くステートのプラグ&プレイボードはCSNが00Hにリセットされる．

・Set RD_DATA Portコマンド　このコマンドはすべてのプラグ&プレイボードのREAD_DATA Portのアドレスをセットする．ライトデータのbits[7:0]がCバスアドレスのbits[9:2]に使用される．Cバスアドレスのbits[1:0]は"11"，bits[15:10]は"000000"固定とする．このコマンドはIsolationステートでのみ使用される．READ_DATA Portの正式な設定方法は次の通りである．

Issue the initiation key
send command Wake[0]
send command Set RD_DATA Port

注意：RESET DRVまたはReset CSNコマンドの後でもこのレジスタは初期化されないので，ソフトウェアで再初期化しなければならない．

- Serial Isolationレジスタ　Serial Isolationレジスタからのリードは，Isolationステートのプラグ&プレイボードをバスリードサイクル（アイソレーション手順参照）に応答させる．切り離されたボードはSleepステートに移行する．
- Card Select Number　Card Select Number(CSN)はそれぞれのボードがアイソレートされ，Isolationステートのボードが1枚になったときユニークにアサインされる．CSN=0は未定義ボードであることを表す．Cバスボードとして有効なCSNは1～255であり，1から順番にアサインされなければならない．CSNはWake[CSN]コマンドによってボードの選択に使用される．RESETまたは，Reset CSNコマンドによりすべてのボードのCSNは0にリセットされる．CSNが0にセットされている時はCSNレジスタは使用しない．
- Wake[CSN]コマンド　このコマンドはSleepステートのボードをIsolationもしくはConfigステートにする．パラメータ00HのWake[CSN]コマンドは，CSNをアサインされていないすべてのボードをIsolationステートにする．00H以外のパラメータのWake[CSN]コマンドはCSNが同じボードのみConfigステートにする．IsolationまたはConfigステートのカードがCSNの一致しないWake[CSN]コマンドを受けたら，Sleepステートに移行する．
  72ビットシリアルIDとリソースデータに関するすべてのプラグ&プレイファンクションは，1つのバイトシリアルデバイスから受ける．このデータのポインタはWake[CSN]コマンドを受信したときはいつでもリセットされる．
- Resource Dataレジスタ　Resource Dataレジスタをリードすることにより Configステートにあるプラグ&プレイボードのリソースデータを1バイト得ることが出来る．リソースデータは常に1バイトずつ返される．Resource Dataレジ

スタをリードする前に必ずリソースデータがリードできるかどうかStatusレジスタをリードして確認しなければならない.

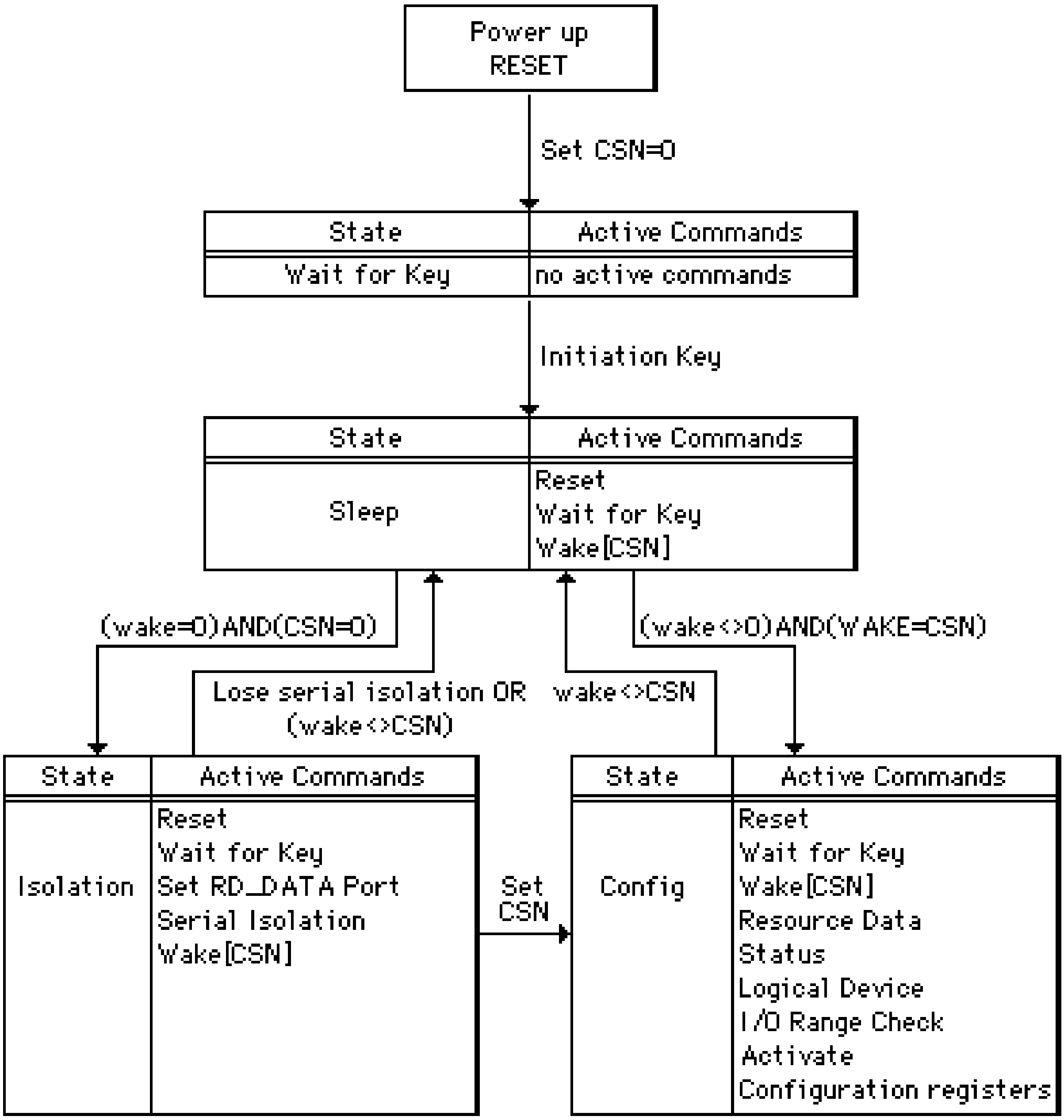
・Statusレジスタ　Statusレジスタのbit[0]はリソースデータの次の1バイトがリードできるかどうかを示している．もしこのビットが1ならリード可能，そうでなければリソースデータはまだ準備されていない．プラグ&プレイソフトウェアはbit[0]がセットされるまで待ち，Resource Dataレジスタから次のデータをリードする.

・Logical Device Numberレジスタ　Logical device Numberレジスタはlogical deviceを選択し，コンフィグレーションコマンドを有効にするために使用される．ボードに1つ以上のlogical deviceが含まれている場合，このレジスタに8ビットのlogical device numberをライトすることにより，logical deviceを選択する．logical device numberはリードしたリソースデータのlogical deviceの順番によって決定する．最初のlogical device numberは0で，続いて1，2…となる.

・I/O Range Checkレジスタ　I/O Range Checkレジスタはlogical deviceにアサインされたI/O portが，他のデバイスと競合していないかをプラグ&プレイソフトウェアが判断するために使用する．I/O range checkはすべてのI/Oレンジについて，I/Oリードコマンドでlogical deviceが55HとAAHを返すことによって実行される．プラグ&プレイソフトウェアはlogical deviceが使用するポートをすべてリードしデータが正しいかどうかチェックする．もし競合が検出されたらプラグ&プレイソフトウェアはlogical deviceのI/Oを再配置する．このレジスタのbit[1]をセットすることでI/O range checkのロジックがイネーブルになる．bit[0]をセットすると，logical deviceはアサインされたI/Oのリードに対し55Hを返す．bit[0]をクリアすると，logical deviceはアサインされたI/Oのリードに対しAAHを返す．このファンクションはActivateレジスタのbit[0]が0の時のみ実行される．このコマンドはオプションであり，コンフィグ可能なI/Oポートを持たないボードにはインプリメントしない.

・Activateレジスタ　Activateレジスタはlogical deviceをアクティブ/インアクティブにするために使用されるリード/ライトレジスタである．アクティブな

logical deviceは通常動作のすべてのCバスサイクルに応答する．インアクティブなlogical deviceは，どのCバス信号もドライブせず応答もしない．bit[0]がアクティブビットで，"1"にセットされればlogical deviceはアクティブになり，そうでなければインアクティブになる．



■プラグ＆プレイアイソレーションの流れ

パワーオン時，すべてのプラグ&プレイボードはRESETを検出し，CSNを0にセットしWait for Keyステートに移行する．初期設定情報を不揮発デバイスからロードするために，プラグ＆プレイポートへのアクセスはRESETやResetコマンドから2msecのウェイトを入れる必要がある．

Wait for Keyステートのボードは，initiation keyを検出するまで自動コンフィグレーションポートへのアクセスに応答しない．ボードはプラグ&プレイインターフェィスへのすべてのCバスアクセスを無視する．

ボードがinitiation keyを受信したらSleepステートに移行する．このステートではボードはライトデータが00HのWake[CSN]コマンドを待つ．このWake[CSN]コマンドはすべてのボードをIsolationステートにし，シリアルID/リソースデータのポインタをリセットする．

Set RD_DATA portコマンドを使用してREAD_DATA port addressをセットする必要があるため，最初にボードはIsolationステートに移行する．

次にボードをアイソレートするためにSerial Isolationレジスタを72組リードする．ボードからリードしたチェックサムが有効であれば1枚のボードをアイソレートしたことを意味する．アイソレートされたボードは他のすべてのボードがアイソレーション手順からはずれ，Sleepステートに戻る間，Isolationステートのままである．ボードのCSNはユニークな番号をセットされる．この値をライトすることによりConfigステートに移行する．Wake[0]コマンドを送ることによりこのボードがSleepステートに戻り，CSNが0であるすべてのボードがIsolationステートに移行する．この一連の処理はプラグ&プレイボードが検出されなくなるまで行われる．もし，READ_DATA Portの競合を検出したら，すべてのカードをIsolationステートにし，シリアルデータポインタをリセットするためにWake[0]コマンドを発行する．ソフトウェアは次のシリアルアイソレーションリードサイクルを始める前に1msecのウェイトを入れなければならない．

■リソースデータのリード

ボードのリソースデータはConfigステートのボードからのみリードされる．ボードは2つの異なった方法のうちの1つによってConfigステートになる．ボードはアイソレーション処理で残り，CSNをアサインされることによりConfigステートに移行する．またボードはCSNに合ったWake[CSN]コマンドを受けることによってもConfigステートに移行する．

72ビットのシリアルIDおよびリソースデータに関するすべてのプラグ&プレイファンクションはシリアルデバイスからリードされる．また，シリアルデバイスのポインタはWake[CSN]コマンドでリセットされる．これはボードがWake[CSN]コマンドによってSleepステートからConfigステートに移行したらリソースデータをアクセスされる前に，まず9バイトのシリアルIDをリードされることを意味する．ベンダIDおよびユニークシリアルナンバは有効であるが，この方法でリードされたチェックサムは無視する．IsolationステートからConfigステートに移行したボード（すなわちアイソレーションプロトコルが実行され，シリアルIDの72ビットをすべてリードされた）はResource Dataレジスタの最初のリードでリソースデータを返す．

ボードのリソースデータはStatusレジスタをポーリングしてbit[0]がセットされるまで待ってからリードされる．Resource Dataレジスタからリソースデータを1バイトリードする準備ができたとき，このビットがセットされる．Resource Dataレジスタをリードしたら，次のリソースデータをリードする前にStatusレジスタをポーリングしなければならない．この処理をすべてのリソースデータをリードするまで行う．

これらの処理はハードウェアがResource Dataレジスタのデータ（8ビット）を準備しておく必要があることを意味する．この処理が完了したらステータスbit[0]はセットされる．Resource Dataレジスタがリードされたら，ステータスbit[0]がクリアされ，Resource Dataレジスタの8ビットが準備できたら，ステータスbit[0]を再びセットする．

■ボードのコンフィグレーション

　プラグ＆プレイボードはlogical device毎に使う標準Cバスリソースを設定するための下記のレジスタをサポートする.

・Memory Address Baseレジスタ(4つまでの非連続領域).
・I/O Address Baseレジスタ(8つまでの非連続領域).
・Interrupt Level Selectレジスタ(2つまでの割り込みレベル).
・DMA Channel Selectレジスタ(2つまでのDMAチャネル).

　これらのレジスタはリード/ライト可能で，常にプラグ＆プレイボードのすべてのlogical deviceの動作に反映する．リソースがプログラマブルでなければ，コンフィグレーションレジスタビットはリードオンリーである.

●コンフィグレーションの順番

　それぞれのプラグ＆プレイボードから，ボード上のlogical deviceのリソース要求について記述されたリソースデータがリードされる．プログラマブルなコンフィグレーションレジスタはボードのプラグ＆プレイインターフェィスを通して設定される.

　logical deviceのコンフィグレーションレジスタは，最初にLogical Device Number Registerにlogical device numberをライトすることによって設定される．logical device numberはリードされたボードのリソースデータのlogical deviceの順番で決定する．最初にリードされたlogical deviceのlogical device numberが0となり，続いて1，2…となる．logical deviceを選択した後，プラグ＆プレイレジスタインターフェィスへ正しい値をライトすることによりコンフィグレーションレジスタが設定される．それぞれのリソースのためにライトされるプラグ＆プレイレジスタは，リソースデータをリードした順番によって決定する．例えば，logical deviceが次のようなリソースデータ構造を持っていたとする.

```
TAG Logical Device
   TAG Memory Descriptor    (becomes memory descriptor 0)
   TAG Memory Descriptor    (becomes memory descriotor 1)
   TAG I/O Descriptor       (becomes I/O descriptor 0)
   TAG Start DF             (dependent function)
      TAG I/O Descriptor    (becomes I/O descriptor 1)
      TAG Int Descriptor    (becomes interrupt descriptor 0)
   TAG Start DF             (next set of dependent resources)
      TAG I/O Descriptor    (becomes I/O descriptor 1)
      TAG Int Descriptor    (becomes interrupt descriptor 0)
   TAG End DF               (end of resource dependencies)
TAG END
```

この例ではlogical deviceは2つの独立したメモリエリアと，1つの独立したI/Oエリアを必要としている．これらはリソースデータをリードした順番に従ってMemory Descriptor 0, Memory Descriptor 1, I/O Descriptor 0と定義される．Memory Descriptor 0はプラグ＆プレイレジスタの40H～44H, Memory Descriptor 1はプラグ＆プレイレジスタの48H～4CH, I/O Despriptor 0はプラグ＆プレイレジスタの60H～61Hにプログラムされる．独立なfunctionはdependent functionより前に置くことを推奨する．

このlogical deviceの次のリソースデータは，1つのI/O portと1つの割り込みのdependent functionを呼び出す．これらはこのlogical deviceからのリードの2番目のI/Oと最初の割り込みのリソースであるから，I/O descriptor 1とinterrupt descriptor 0となる．次のdependent function entryもI/O descriptor 1とinterrupt descriptor 0を示している．

これらのレジスタは任意の順番でプログラムすることができる．

●リソースプログラミング

プラグ＆プレイボードは，ボードのCSNをライトデータとしたWake[CSN]コマン

ドを送ることによりプログラミングされる．これでCSNが合った1枚のボードのみがConfigステートになり，他のすべてのボードはSleepステートになる．次にLogical Device Numberレジスタにlogical device numberをライトすることによりlogical deviceを選択する．ボードのlogical deviceが1つしか無いときはこの処理をする必要はない．

logical device毎のリソースコンフィグレーションをボードのI/O，メモリ，IRQ，DMAのレジスタを使ってプログラミングする．たとえリソースがアサインされなくてもlogical deviceのリソース要求はプログラムしなければならない(ディセーブルになる値をライトする)．それぞれのリソースタイプは次の通りである．

・Memory Configuration　メモリ空間リソースはmemory base addressレジスタにmemory base addressをライトすることによりプログラミングされる．次に，memory controlに適正な8/16/32ビットメモリ動作とデコードセレクトオプションをライトする．もし，メモリデコードオプションにレンジ長をセットするときは，レンジ長をmemoy upper limit/range lengthレジスタにライトする．最高限をセットするときは，最高限度アドレスをmemoy upper limit/range lengthレジスタにライトする．メモリリソースのアサインが無い場合はmemory base addressレジスタおよびupper limit/range lengthレジスタには0をセットする．

・I/O Space Configuration　I/O空間リソースは，I/O port base addressレジスタにI/O base address[15:00]をライトすることによりプログラミングされる．I/Oリソースのアサインが無い場合はI/O base addressレジスタに0をセットする．

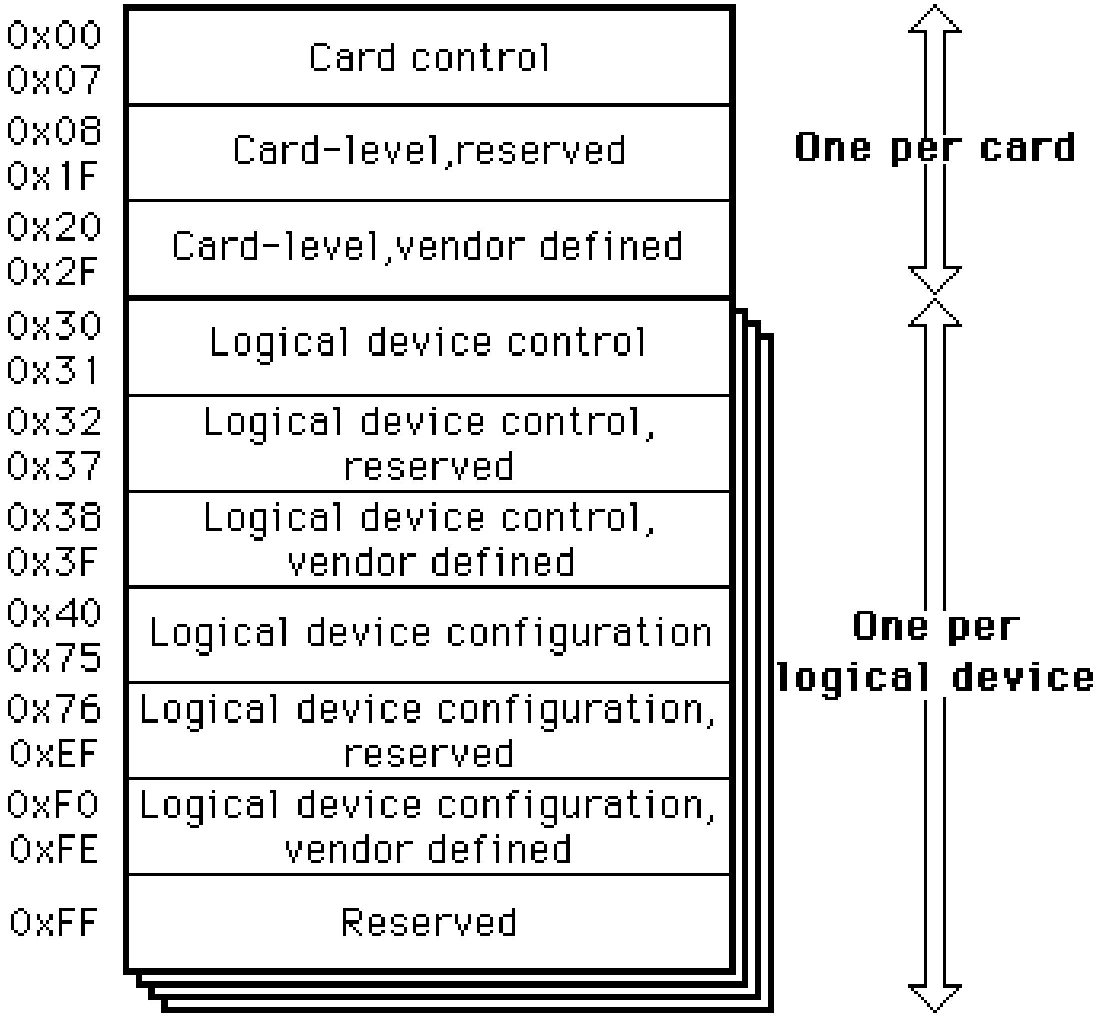
・Interrupt Request Level　logical deviceのinterrupt request levelはInterrupt Level Selectレジスタにinterrupt request level numberをライトすることによって選択される．このselect numberは，IR numberを意味する．割り込みのedge/levelおよびhigh/lowアクティブはInterrupt Request Typeレジスタにライトしなければならない．割り込みをアサインしない場合はInterrupt Level Selectレジスタに0をセットする．

・DMA Channel　logical deviceのDMA channelは，DMA Channel Selectレジスタに DMA channel numberをライトすることにより選択される．select numberはDRQ/DACK channelを意味する．DMA channelをアサインしない場合はこのレジスタに4をライトする．
最後にlogical deviceのアクティブビットをセットする．これでバス上のリソースをアサインされたlogical deviceがアクティブになる．コンフィグレーションレジスタのプログラミングが終わったら，すべてのボードはWait for Keyステートにセットされなければならない．

3.5　プラグ＆プレイ標準レジスタ

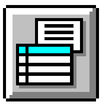
次の図はcardレジスタとlogical deviceレジスタの順序を表している．このようにcardレジスタはボード毎に，logical deviceレジスタはボードのlogical device毎にある．

注意：00H～2FHでインプリメントされていないレジスタはリードで0を返さなければならない．



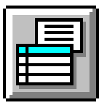
■プラグ＆プレイCard Control Registers

▼*プラグ&プレイカードコントロールレジスタ*



■プラグ＆プレイLogical Device Control Registers

▼*プラグ&プレイLogical Device Control Difinition*



■プラグ＆プレイLogical Device Configuration Registers

プラグ&プレイボードのリソースアサインはコマンドの発行と同様に，アドレスおよびコマンド/データポートを通してプログラミングされる．次のレジスタはプラグ&プレイボードが使用するメモリ，I/O，割り込み，DMAソースを定義する．これらのコンフィグレーションレジスタはすべてカレントのlogical deviceのものである．

●Memory Configuration Registers

次表はメモリリソースの設定方法を規定する．もしコンフィグレーションソフトウェアが何らかの理由でメモリアロケートできない場合はメモリリソースレジスタ(例えば40H，41H，43H，44H)に0をライトする．これによりメモリレンジはディセーブルになる．

logical device毎に最大4つのメモリブロックを持てる．

memory range lengthレジスタはアドレスbits[23:08](32ビットメモリの場合は[31:00])のマスクで定義されている．マスクビットがセットされていたらメモリアドレスのそのビットを比較に使用する．クリアされていたらそのビットは比較しない．例えばlogical deviceが64KBのメモリを要求している場合は次のようにする．

memory range length[23:16]=FFH
memory range length[15:08]=00H

これはアドレスのbits[23:16]のみを比較に使用し，bits[15:08]およびbits[07:00]は比較しないことを意味する．この64KBアドレス領域は64KB以上のバウンダリでなければならない．

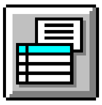
memory upper limitレジスタはメモリリソースの割り当てよりも1バイト大きい値が定義される．例えば，64KByteのメモリを要求しているlogical deviceに1MByteのbase addressをアサインしたときは次のようになる．

memory base[23:16]=10H
memory base[15:08]=00H
memory upper limit[23:16]=11H
memory upper limit[15:08]=00H

memory controlレジスタ42H(32ビットメモリの場合は7AH)はプログラマブルな8/16ビット動作をサポートせずデコードタイプが1種類しか使用できないデバイスではリードオンリーでもよい．43Hおよび44H（32ビットメモリの場合は7BH～7EH）のmemoryレジスタは自然なバウンダリでの固定長メモリ領域のみをサポートするデバイスではサポートする必要はない．

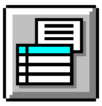
24ビットと32ビットmemory descriptorの混在は許されない．リソースデータの最初のmemory descriptorのタイプによってコンフィグレーションレジスタは40H～5FHか46H～A8Hのどちらを使用するか決定する．

▼*メモリスペースコンフィグレーション*



インプリメントされていないmemory configuration registerはリード時に0を返さなければならない．

▼*32ビットメモリスペースコンフィグレーション*

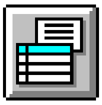


インプリメントされていない32ビットmemory configuration registerはリード時に0を返さなければならない．

●I/O Configuration Registers

コンフィグレーションレジスタの60H～6FHはI/Oコンフィグレーションに使用される．logical device毎に最大8つのI/O descriptorを持てる．ベースアドレスを0000HにするとI/Oレンジはディセーブルになる．

▼*I/0コンフィグレーションレジスタ*

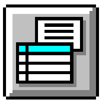


インプリメントされていないI/O configuration registerはリード時に0を返さなければならない．

●Interrupt Configuration Registers

プラグ&プレイボードのそれぞれのlogical deviceは2つまでの割り込み要求を使用することができる．

▼*Interruptコンフィグレーションレジスタ*



インプリメントされていないinterrupt configuration registerはリード時に0を返さなければならない．

●DMA Configuration Registers

プラグ&プレイボードのそれぞれのlogical deviceは2つまでのDMAチャネルを使用することができる．

▼DMAコンフィグレーションレジスタ

インプリメントされていないDMA configuration registerはリード時に4を返さなければならない.

●Reserved and Vendor Defined Configuration Registers

レジスタA9H～EFHは将来のプラグ＆プレイ拡張のためにリザーブされている.レジスタF0H～FEHはベンダ定義とする.

Name	Register Index	Definition
Logical device configuration reserved	A9H-EFH	*reserved*
Logical device configuration vendor defined	F0H-FEH	*vendor defined*

■Reserved Register

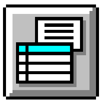
レジスタFFHはリザーブされている.

Name	Register Index	Definition
Reserved	FFH	*reserved*

3.6　プラグ＆プレイリソース

■シリアルID

▼*シリアルID*



●Vendor ID

NESAのメーカIDを使う．ただしMSBを"1"にする．データは上位8ビット，下位8ビットの順に並べる．

●Product Number

ベンダがアサインする．ボード毎に共通．データは上位8ビット，下位8ビットの順に並べる．最下位4ビットはレビジョンナンバ．

●Serial/Unique Number

32ビットserial numberは独立のプラグ＆プレイボードを選択するためのアイソレーション処理にのみ使用する．これは1つのシステムに複数の同一vendor ID/Product Numberのボードを差したときに区別するために使用する．従ってユニークでなければならない．もしこの特徴をサポートしないときは"FFFFFFFF"を返さなければならない．この場合システム内でvendor ID/Product Number毎に1枚のボードしかサポートできないことを意味する．

●Checksum

checksumはデバイス認識情報をリードするときに発生する競合を確実になくすために使用される．

■プラグ＆プレイリソースデータタイプ

プラグ&プレイリソースデータはプラグ&プレイボードのすべてのリソース要求を完全に記述している．プラグ&プレイリソースデータは”タグ”データ構造の並びで与えられる．プラグ&プレイボードには最低限2つの違ったタイプのデータを必要とする．これらをsmall item，large itemと呼ぶ．構造の一般的なフォーマットを次に示す．第1バイトはタイプやサイズなどを定義し，それ以降に実際の情報が続く．第1バイトのbit[7]をsmall itemかlarge itemの区別に使用する．

●リソースデータ要求

プラグ&プレイボードのリソースデータには，常にプラグ＆プレイVersion Number，Identifier Stringおよび固定リソースを区別するためのすべてのリソースタイプが存在する．Identifier StringはOSがデバイス関連コンフィグレーションおよび，エラーメッセージに使用する．ボードベンダは複数のテキストフォーマット(ANSIまたはSHIFT-JIS，UNICODE)および外国語をサポートするなら複数のIDを入れることができる．

プラグ&プレイLogical Deviceは，任意の数のリソースとsmallおよびlarge item data typeの任意の組み合わせを使用することが出来る．一般形式は次の通り．

1. プラグ＆プレイVersion Number.
2. Identifier String.
3. Logical Device ID.
    - このLogical Deviceの任意の互換デバイスID
    - このLogical Deviceが使用するリソース(IRQ，メモリ，I/O，DMA)

    (3はLogical Device毎に繰り返される）.
4. リソースの終わりを示すEnd Tag.

コンフィグレーションレジスタはリソースのリードと同じ順番で設定されるため

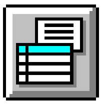
リソース記述の順番は重要である(ボードのコンフィグレーションを参照). さらにDependent Functionの場合にはDependent Functionの設定にかかわらずdescriptorとレジスタの配置を一定にするためにnull descriptorを含める必要があるかもしれない.

null descriptorはそれぞれのリソースタイプで定義される.

●Small Resource Data Type

small resource data typeはサイズが2～8バイトで次のようなフォーマットになる.

▼*Small Resource Data Type Tag Bit*



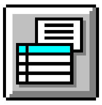
Small Resource Item Names

Small Item Name	Value
Plug and Play Version Number	0x1
Logical Device ID	0x2
Compatible Device ID	0x3
IRQ Format	0x4
DMA Format	0x5
Start Dependent Function	0x6
End Dependent Function	0x7
I/O Port Descriptor	0x8
Fixed Location I/O Port Descriptor	0x9
Reserved	0xA-0xD
Vendor Defined	0xE
End Tag	0xF

・プラグ＆プレイVersion Number

プラグ＆プレイversion numberはそのボードと互換性のあるプラグ＆プレイスペックのバージョンを示す．ベンダ特有のバージョンナンバも含まれ，ボードのバージョンをデバイスドライバが確認するために使用される．

▼ *プラグ＆プレイVersion Number*



・Logical Device ID

Logical Device IDは1枚のボード上の複数のlogical deviceを識別するために使用する．Logical Device IDのフォーマットは，シリアルIDのVendor ID+

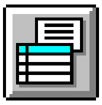
Product Numberと同様である.

bits[15:0]　　Vendor ID bits[15:0]. NESAのメーカIDを使う. MSBを"1"にする. Logical Device IDのbits[7:0]がVendor IDのbits[15:8], Logical Device IDのbits[15:8]がVendor IDのbits[7:0]にあたる.

bits[31:16]　　ベンダがアサインするデバイスコード. 1枚のボード上のそれぞれのlogical deviceに関してユニークな値にしなければならない.

このIDはデバイスドライバの選択に使用される. またオプションのコマンドをサポートしているかどうかの情報を含んでいる. flag field 0のbit[0]が"1"ならばブート時にBIOSによってアクティブになることを意味する.

▼*Logical Device ID*



・Compatible Device ID

compatible device IDは互換性のある他のデバイスのlogical device IDを持つ. 必要ならば0Sはこの情報を互換性のあるデバイスドライバのロードに使用する. それぞれのデバイスに対し, 複数のcompatible device IDを置くことができる.

Offset	Field Name
Byte 0	0001100B (Type=0,small item name=0x3,length=4)
Byte 1	compatible device ID bits[7:0]
Byte 2	compatible device ID bits[15:8]
Byte 3	compatible device ID bits[23:16]
Byte 4	compatible device ID bits[31:24]

compatible IDの使用例で，logical IDがABCD0000Hのデバイスを出荷しているボードベンダを考える．後にlogical IDがABCD0001Hの新しいデバイスを出荷する．この新しいデバイスは古いデバイスと100%の互換性を持ち，さらに機能が追加されている．この場合，見つかればABCD0001Hのドライバをロードする．ABCD0001Hのドライバが見つからなければABCD0000Hのドライバをロードする．

・IRQ Format

Null IRQ Descriptor: IRQ mask bitをすべて0にする(他のフィールドは無視)．

プラグ&プレイソフトウェア動作: 該当するすべてのIRQ level select registerに0をセットする．IRQ type select registerには02H(high edge sense)をセットする．

Offset	Field Name
Byte 0	0010001xB (Type=0,small item name=0x4,length=2 or 3)
Byte 1	IRQ mask bits[7:0] bit[0]がIRQ0を示す。1で要求。
Byte 2	IRQ mask bits[15:8] bit[0]がIRQ8を示す。1で要求。
Byte 3[1]	bit[7:4] reserve(0) bit[3] low level sense bit[2] high level sense bit[1] low edge sense bit[0] high edge sense(Cバスコンパチ)

1　Cバスではbit[0]のみ1とする

・DMA Format

Null DMA Descriptor: DMA channel mask bitをすべて0にする(他のフィー

ルドは無視).

　プラグ＆プレイソフトウェア動作: 該当するすべてのDMA channel select registerに04Hをセットする.

Offset	Field Name
Byte 0	00101010B (Type=0,small item name=0x5,length=2)
Byte 1	DMA channel mask bits[3:0] bit[0]がchannel0を示す。1で要求 bits[7:4] reserved
Byte 2	bits[1:0] DMA転送タイプ 00 8bit 01 8 or 16bit 10 16bit 11 reserve bit[2] logical device bus master 0 not bus master bit[3] DMA byte mode 0 not execute 1 execute(DMA転送タイプが8bitの時) bit[4] DMA word mode 0 not execute 1 execute(DMA転送タイプが16bitの時) bits[6:5] DMA channel speed 00 compatible(Cバスコンパチ) bit[7] reserve(0)

・Start Dependent Function

　それぞれlogical deviceはリソースのセットを要求する. このリソースのセットはアービトレーションをするために相互に関連があることがある. このような関連しているfunctionのデータ構造定義には次のものを使用する.

Offset	Field Name
Byte 0	0011000xB (Type=0,small item name=0x6,length=0 or 1)

start dependent function fieldは0または1バイト長になる．追加バイトはstart DF tagに続くリソースグループに優先順位を付けるために使用する．追加バイトが無いときには優先順位は無い．

追加バイト

Value	Field Name
0	最適設定
1	設定可能
2	設定可だが望ましくない
3-255	*Reserved*

・End Dependent Function

Offset	Field Name
Byte 0	00111000B (Type=0,small item name=0x7,length=0)

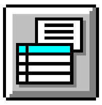
End Dependent Functionはlogical device毎に1つしか許されない．つまりDependent Functionはネスト出来ない．

・I/O Port Descriptor

I/Oポートの記述には2つのタイプがある．1つは完全にプログラマブルなCバスボード．もう1つは固定I/Oが必要な古いタイプのCバスボードである．プラグ&プレイ対応ボードはすべて16ビットデコードとするので固定I/Oリソース要求にもI/O Port Descriptorをrange minimum base address=range maximum base addressで使用する．Fixed I/O Location I/O Port Descriptorは10ビットデコードのみのサポートであるためPC-98では使用禁止である．

Null I/O Port Descriptor：Range length fieldをすべて0にする(他のフィールドは無視)．
プラグ&プレイソフトウェア動作：該当するすべてのI/O port base address registerに0をセットする．

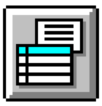
▼*I/O Port Descriptor*



minimum base addressはalignment fieldのバウンダリ(整数倍)とする．

・Fixed Location I/O Port Descriptor

▼*Fixed Location I/O Port Descriptor*



・Vendor Defined

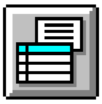
これはベンダが自由に使える．

Offset	Field Name
Byte 0	01110xxxB (Type=0,small item name=0xE,length=1-7)
Byte 1 to 7	vendor defined

・End Tag

End Tagはリソースデータの終わりを意味する．このfieldのbyte 1が0の時はchecksumは正しいものとして取り扱う．

▼*End Tag*



●Large Resource Data Type

次に示すように大きなコンフィグレーションデータ構造を含むデータタイプを許す．これは64Kバイトまでのデータを許す16bit length fieldを含む．

Large Resource Data Type Tag Bit

Offset	Field Name
Byte 0	1xxxxxxxB (Type=1,large item name=xxxxxxx)
Byte 1	データアイテム長 bits[07:00]
Byte 2	データアイテム長 bits[15:08]
Bytes 3 to n	データアイテム

Large Resource Item Names

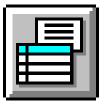
Large Item Name	Value
Memory Range Descriptor	0x1
Identifier String(ANSI)	0x2
Identifier String(Unicode)	0x3
Vendor Defined	0x4
32-bit memory range descriptor	0x5
32-bit fixed location memory range descriptor	0x6
Reserved	0x5-0x7F

・Memory Range Descriptor

Null Memory Range Descriptor：Range length fieldをすべて0にする(他のフィールドは無視).

プラグ&プレイソフトウェア動作：該当するすべてのbase address registerとlimit address/range length registerに0をセットする.

▼*Memory Range Descriptor*



base alignmentの最小値を256とする．minimum base addressはalignment fieldのバウンダリ(整数倍)とする.

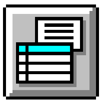
アドレスのbits[7:0]はすべて0とする．またrange minimum base address= range maximum base addressとすることで，固定メモリアドレスのdescriptorとして使用できる.

24ビットと32ビットmemory descriptorの混在は禁止する.

　base addressに100000H以下を指定する場合はD0000H～DFFFFHを優先的に使用するようにする.

・ANSI Identifier String

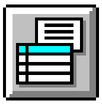
▼*ANSI Identifier String*



　identifier stringはASCIIまたはSHIFT JISコードによって表される．文字列長はヘッダに含まれるので’0’で終端する必要はない．それぞれのボードにはidentifier stringが必要だが，それぞれのlogical deviceには任意である.

・Unicode Identifier String

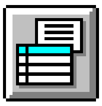
▼*Unicode Identifier String*



　通常はANSI identifier stringを使用する．UNICODEでの定義は後に行なう.

・Vendor Defined

▼*Vendor Defined*

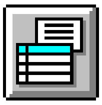


・32ビットMemory Range Descriptor

Null Memory Range Descriptor： Range length fieldをすべて0にする(他のフィールドは無視).

プラグ&プレイソフトウェア動作： 該当するすべてのbase address registerとlimit address/range length registerに0をセットする.

▼32-bit Memory Range Descriptor

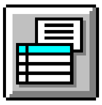


minimum base addressはalignment fieldのバウンダリ(整数倍)とする.

24ビットと32ビットmemory descriptorの混在は禁止する.

・32ビットFixed Location Memory Range Descriptor

▼32ビットFixed Location Memory Range Descriptor



■リソースデータの作成方法

Vendor ID	Byte 0
	Byte 1
	Byte 2
	Byte 3
Serial Number	Byte 0
	Byte 1
	Byte 2
	Byte 3
Checksum	Byte 0
Plug and Play Version and Card String ID	Byte 0
Logical Device ID 0	Byte 0
Resources for Device ID 0	Byte 0
	Byte 1
- -	
	Byte m
Logical Device ID x	Byte 0
Resources for Device ID x	Byte 0
	Byte 1
- -	
	Byte n
End Tag	Byte 0

●サンプルボード

ROM	C0000h～DFFFFhのうちの4KB
I/O	C000h～EFFFhのうちの2ポート
INT	0,1,2,3,5,6(IR 3,5,6,9,12,13)のいずれか
DMA	0,2,3のいずれか
非ブートデバイス	
Vendor ID	B8A3h
Product Number	0001h
Serial Number	01234567h
Logical Device ID	B8A30001h

リソースデータ

```
;
;プラグ＆プレイHeader
;
DB B8H                  ;Vendor ID
DB A3H
DB 00H                  ;Product Number
DB 01H
DB 01H                  ;Serial Number
DB 23H
DB 45H
DB 67H
DB DFH                  ;Checksum
;
;プラグ＆プレイVersion
;
```

```
DB 0AH                  ;プラグ&プレイVersion Number Tag
DB 10H                  ;Version 1.0
DB 10H                  ;Vendor Specific Version Number
;
;ANSI Identifier String
;
DB 82H                  ;ANSI Identifier String Tag
DB 0CH                  ;String Length(12)
DB 00H
DB "Sample Board"       ;ANSI Identifier String
;
;Logical Device ID
;
DB 15H                  ;Logical Device ID Tag
DB B8H                  ;Logical Device ID
DB A3H
DB 00H
DB 01H
DB 00H                  ;Not Required for Boot
;
;I/O Port Descriptor
;
DB 47H                  ;I/O Port Descriptor Tag
DB 01H                  ;Information(16bit Decode)
DB 00H                  ;Range Minimum Base Address(C000H)
DB C0H
DB FEH                  ;Range Maximum Base Address(EFFEH)
DB EFH
DB 02H                  ;Base Address Increment(2)
DB 02H                  ;Number of Ports Required
;
;IRQ Format
;
DB 23H                  ;IRQ Format Tag
```

```
DB 68H                    ;IR Level(3,5,6,9,12,13)
DB 32H
DB 01H                    ;Information(High True,Edge)
;
;DMA Format
;
DB 2AH                    ;DMA Format Tag
DB 0DH                    ;DMA Channel(0,2,3)
DB 00H                    ;DMA Mode
;
;Memory Range Descriptor
;
DB 81H                    ;Memory Range Descriptor Tag
DB 09H                    ;Length(9)
DB 00H
DB 50H                    ;Information
DB 00H                    ;Range Minimum Base Address(0C0000H)
DB 0CH
DB F0H                    ;Range Maximum Base Address(0DF000H)
DB 0DH
DB 00H                    ;Base Address Increment(4KByte)
DB 10H
DB 10H                    ;Range Length(16Blocks=4KByte)
DB 00H
;
;End Tag
;
DB 79H                    ;End Tag
DB 03H                    ;Checksum of All Resource Data
```

●Dependent Functionの例

・例1

最初の例は優先順位が無い単純なボードである．この例のボードはBase Addressが0088H，0188Hまたは0288Hの連続した4つのI/Oポートと，IRQ 3，10，12，13のうちの1つの割り込みを必要とする．IRQの選択とI/Oポートの選択は完全に分離されている．このボードのリソースデータは次のようになる．

```
TAG プラグ＆プレイversion Number
TAG Identifier String
TAG Logical Device ID
TAG IRQ Format (MASK IRQ 3,10,12,13)
TAG Start DF (Lenghth 0)
    TAG I/O Format (Length 4,Min Base 0x0088,Max Base 0x0088)
TAG Start DF (Lenghth 0)
    TAG I/O Format (Length 4,Min Base 0x0188,Max Base 0x0188)
TAG Start DF (Lenghth 0)
    TAG I/O Format (Length 4,Min Base 0x0288,Max Base 0x0288)
TAG End DF
TAG END Tag
```

・例2

Dependent Functionはリソース間の相互依存の関係を記述することができる．例えば，I/Oポートの00D0HとIRQ 3またはI/Oポートの01D0HとIRQ 10の組み合わせで設定したい時のリソースデータは次のようになる．

```
TAG プラグ＆プレイversion Number
TAG Identifier String
TAG Logical Device ID
TAG Start DF (Lenghth 0)
```

```
        TAG I/O Format (Length 4,Min Base 0x00D0,Max Base 0x00D0)
        TAG IRQ Format (MASK IRQ 3)
    TAG Start DF (Lenghth 0)
        TAG I/O Format (Length 4,Min Base 0x01D0,Max Base 0x01D0)
        TAG IRQ Format (MASK IRQ 10)
    TAG End DF
    TAG END Tag
```

・例3

Dependent Functionはまた，設定に優先順位を付けて記述することもできる．この例ではメモリ空間を使用できるなら，ボードはメモリバッファを使用してデータ転送を実行する．メモリ空間を使用できなければボードはDMAチャネルを使用する．また，0300H～03F0Hのどこかの連続した16個のI/Oポートと，IRQ 3，5，6，10，12のうちのどれか1つの割り込みを必要とする．128KByteのメモリウィンドウが必要であるが，使用できなければ64Kバイト，全く使用できなければDMAチャネルを使用する．DMAはチャネル0および3をサポートする．このデバイスのリソースデータは次のようになる．

```
    TAG プラグ＆プレイversion Number
    TAG Identifier String
    TAG Logical Device ID
    TAG I/O Format (Length 16,Min Base 0x0300,Max Base 0x03F0)
    TAG IRQ Format (MASK IRQ 3,5,6,10,12)
    TAG Start DF (Lenghth 1) (Priority 0)
        TAG Memory Format (Length=128K,Min Base 0x0C0000,
                                       Max Base 0x0E0000)
        TAG DMA Format (MASK no DMA channel)
    TAG Start DF (Lenghth 1) (Priority 1)
        TAG Memory Format (Length=64K,Min Base 0x0C0000,
                                      Max Base 0x0E0000)
```

```
    TAG DMA Format (MASK no DMA channel)
TAG Start DF (Lenghth 1) (Priority 2)
    TAG Memory Format (Length=0,Min Base 0x0C0000,
                                Max Base 0x0E0000)
    TAG DMA Format (MASK 0,3)
TAG End DF
TAG END Tag
```

・例4

次の例はDependent Functionの誤った使い方を示している．これは2つのEND DFを含んでいるが，そのような使い方はできない．

```
TAG プラグ＆プレイversion Number
TAG Identifier String
TAG Logical Device ID
TAG IRQ Format (MASK IRQ 3,10,12,13)
TAG Start DF (Lenghth 0)
    TAG I/O Format (Length=4,Min Base 0x0088,Max Base 0x0088)
TAG Start DF (Lenghth 0)
    TAG I/O Format (Length=4,Min Base 0x0188,Max Base 0x0188)
TAG Start DF (Lenghth 0)
    TAG I/O Format (Length=4,Min Base 0x0288,Max Base 0x0288)
TAG End DF
TAG Start DF (Lenghth 1) (Priority 1)
    TAG DMA Format (MASK no DMA channel)
TAG Start DF (Lenghth 1) (Priority 2)
    TAG DMA Format (MASK 0,3)
TAG End DF
TAG END Tag
```

正しくは次のようになる.

```
TAG プラグ＆プレイversion Number
TAG Identifier String
TAG Logical Device ID
TAG IRQ Format (MASK IRQ 3,10,12,13)
TAG Start DF (Lenghth 0)
    TAG I/O Format (Length=4,Min Base 0x0088,Max Base 0x0088)
    TAG DMA Format (MASK no DMA channel)
TAG Start DF (Lenghth 0)
    TAG I/O Format (Length=4,Min Base 0x0188,Max Base 0x0188)
    TAG DMA Format (MASK no DMA channel)
TAG Start DF (Lenghth 0)
    TAG I/O Format (Length=4,Min Base 0x0288,Max Base 0x0288)
    TAG DMA Format (MASK no DMA channel)
TAG Start DF (Lenghth 1) (Priority 2)
    TAG I/O Format (Length=4,Min Base 0x0088,Max Base 0x0088)
    TAG DMA Format (MASK 0,3)
TAG Start DF (Lenghth 1) (Priority 2)
    TAG I/O Format (Length=4,Min Base 0x0188,Max Base 0x0188)
    TAG DMA Format (MASK 0,3)
TAG Start DF (Lenghth 1) (Priority 2)
    TAG I/O Format (Length=4,Min Base 0x0288,Max Base 0x0288)
    TAG DMA Format (MASK 0,3)
TAG End DF
TAG END Tag
```

3.7　Option ROM Specification

■概要

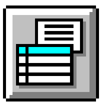
この節ではブートデバイスのプラグ&プレイOption ROMについて述べる．ブートデバイスはOSをロードする前に初期化されなければならない．通常，ブートデバイスとはOSをストアするIPL(Initial Program Load)デバイスである．しかし，ここでは拡張して，重要な入力および出力デバイスも含む．

■プラグ＆プレイOption ROM

Option ROMを持つすべての新しいプラグ＆プレイデバイスはプラグ＆プレイOption ROM Headerをサポートする．加えて，旧デバイスがプラグ＆プレイOption ROM Headerをサポートするためにアップグレードするかもしれない．プラグ&プレイOption ROM Headerは旧デバイスがソフトウェアコンフィグレーション可能なリソースを持たない時は，ブートデバイスの識別や選択をするプラグ&プレイSystem BIOSの手助けとなる．

●Option ROM Header

▼*Option ROM Header*



Option ROM ID- オプションROMの存在を確認するためのID．

各エントリについては，この後の，「本体BIOSとのインターフェイス」を参照すること．

●Expansion Header for プラグ＆プレイ

Offset	Length	Value	Description
00H	DWORD	$PnP(ASCII)	Signature
04H	BYTE	(01H)	Header Revision
05H	BYTE	Varies	Header Length (16byte単位)
06H	WORD	Varies	Offset of next Header (0000Hの時はなし)
08H	BYTE	00H	Reserved
09H	BYTE	Varies	Checksum
0AH	DWORD	Varies	Device Identifire
0EH	WORD	Varies	Pointer to Manufacturer String (Optional)
10H	WORD	Varies	Pointer to Product Name String (Optional)
12H	3 BYTE	Varies	Device Type Code
15H	BYTE	Varies	Device Indicators
16H	WORD	Varies	Boot Connection Vector
18H	WORD	Varies	Disconnect Vector
1AH	WORD	Varies	Bootstrap Entry Point (0000Hの時はなし)
1CH	WORD	0000H	Reserved
1EH	WORD	Varies	Static Resource Information Vector (Optional)

Signature- すべてのExpansion HeadersはユニークなExpansion Header Identifierを含む．プラグ&プレイのExpansion Header IdentifierはASCII文字列の"$PnP"(24H,50H,6EH,50H)である．

Header Revision- このストラクチャのレビジョン番号のみ示し，準拠するプラグ&プレイBIOSのレビジョンを示すものではない．

Offset of next Header- これはこのOption ROMから次のExpansion ROM Headerへリンクするための領域である．もし他のExpansion ROM Headerがなければここは0000Hになる．

Checksum- 各Expansion Headerは個別にチェックサムを持つ．これは有効なExpansion Headerの使用を決定できるソフトウェア(ここではシステムBIOS)を許す．算出方法は8ビットの値としてバイト単位にChecksumを含むすべての拡張ヘッダを合計した値が00Hになるようにする．

Device Identidier- プラグ&プレイLogical Device ID.

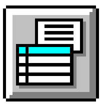
Pointer to Manufacturer String- これはオプションROM内でASCIIZ（ASCIIコードで表し，終わりに00Hを置く）で表されたボードベンダの文字列のポインタへのオフセットである．これはオプションであり，0000Hの時はサポートされていないことを意味する．

Pointer to Product Name String- これはオプションROM内でASCIIZで表されたプロダクト名の文字列のポインタへのオフセットである．これはオプションであり，0000Hの時はサポートされていないことを意味する．

Device Type Code- このフィールドはユーザやOSがブートデバイスを特に指定していない時，システムBIOSがブートデバイスに優先度を付けるのを助けるための一般的なデバイスタイプの情報を含む．

Device Type Codeは3つのバイト領域に分けられる．1バイト目は一般的なデバイスタイプを表すBase-Type Codeからなる．2バイト目はDevice Sub-Typeで定義はBase-Type Codeに従属する．3バイト目はBase-TypeとSub-TypeをもとにDevice Programming Interface(If-type)を定義する．

▼*Device Type Cord*



Device Indicators- このフィールドでDisplay，Input，IPLの3つのデバイスの1つであることを識別できる．

Bit	Description
7	=1の時、Device Driver Initialization Model をサポートする
6	=1の時、このROMはShadow 可能である
5	=1の時、このROMはcacheableである
4	=1の時、このROMはブートデバイスとして選択された時のみ有効であればよい。=0の時、それ以外
3	Reserved(=0)
2	=1の時、IPLデバイスである表す
1	=1の時、Inputデバイスであることを表す
0	=1の時、Displayデバイスであることを表す

以下のエントリはPC-9800シリーズでは使用しない.

Boot Connection Vector
Disconnect Vector
Bootstrap Entry Point
Static Resource Information Vector

●本体BIOSとのインターフェィス

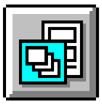
・BIOSから見たプラグ＆プレイ対応オプションROMの初期化方法

初期化エントリ0

初期化エントリ0では, XROM Control IDの設定を必ず行う必要がある. これは続く初期化エントリ1, 拡張エントリおよびブートエントリが呼び出されるための重要な手続きである.

Control IDは，次のように設定する．

▼*Control ID*



初期化エントリ0が呼び出されたとき，DS:BXレジスタにはControl IDを設定するアドレスが格納されている．従って，各XROMでは次のようにControl IDを設定すればよい．

```
mov    byte ptr ds:[bx], 11xxxxxxb
```

8KBのROMの場合はさらに，

```
mov    byte ptr ds:[bx + 1], 10xxxxxxb
```

とし，4KB毎にBXレジスタに1を加えたアドレスにControl IDを設定する．

Control IDを設定した後，XROMで制御するデバイス，メモリ等の初期化を行いRETF命令で処理を終了する．またプラグ&プレイ対応デバイスは，デバイスのコンフィグレーションを行う．この際，DSレジスタを保証しなければならない．

注意：もし，プラグ&プレイ対応拡張ROMがデバイスのコンフィグレーションに失敗した場合は，該当するROM領域のControl IDを全て0にする．

さらに，初期化エントリ0は次のパラメータを受け取る．

ES:DI	BIOSプラグ&プレイInstallation Check Structureのポインタ
CS	Cバスプラグ&プレイのCSN
DX	Cバスプラグ&プレイのRead DATA Port

初期化エントリ1

　このエントリは，初期化時に低速な装置にI/Oを実行するXROMのためのものである．初期化エントリ0でI/Oを実行し，その終了を待たずにExitした時，このエントリでそのI/Oの終了を待つことができる．これによって，複数のデバイスに対して同時にI/Oを実行でき，システムの初期化処理を高速化することができる．

　初期化エントリ1は次のパラメータを受け取る

ES:DI	BIOSプラグ＆プレイInstallation Check Structureのポインタ
CS	Cバスプラグ＆プレイのCSN
DX	Cバスプラグ＆プレイのRead DATA Port

注意：コンフィグレーションに失敗したプラグ＆プレイ対応デバイス（Control ID = 0）のエントリは呼び出さない．

拡張エントリ

　このエントリは，システム初期化処理やブート処理の拡張のために用意されている．必要ない場合は，このエントリにRETF命令を用意し，直ちに処理を終了すればよい．

　拡張エントリは次のパラメータを受け取る

ES:DI	BIOSプラグ＆プレイInstallation Check Structureのポインタ
CS	Cバスプラグ＆プレイのCSN
DX	Cバスプラグ＆プレイのRead DATA Port

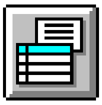
ブートエントリ

　このエントリは，上位プログラム（IPL）のロードを行うためのものである．

　本体BIOSは，ブートプライオリティをALレジスタにセットし，このエントリを呼び出す．各拡張ROMは該当するブートプライオリティ値を得たときのみ，ブート処理を試みる．その他の値の場合は，直ちにRETF命令で本体BIOSへ制御を戻す．

また，ブートに失敗した場合もRETF命令で処理を終了する．

▼ブートプライオリティ



　上記以外のブートプライオリティは未使用であるが，将来の拡張のために次のように予約されている．

プライオリティ	デバイス
0000	標準
0001～1000	FD系デバイス
1001	回線系デバイス
1010～1111	固定ディスク系デバイス

注意：コンフィグレーションに失敗したプラグ＆プレイ対応デバイス（Control ID = 0）のエントリは呼び出さない．

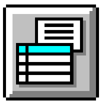
　さらに，ブートエントリは次のパラメータを受け取る

ES:DI　BIOSプラグ&プレイInstallation Check Structureのポインタ
CS　　Cバスプラグ&プレイのCSN
DX　　Cバスプラグ&プレイのRead DATA Port

・インストールチェック構造体
　F0000H～FFFFFHの空間内で16バイトバウンダリ毎に文字列"$PnP"をサーチする．文字列"$PnP"があればプラグ＆プレイBIOSが存在しており，次のような構造

体から必要な情報が取得できる.

▼インストールチェック構造体



Signature - byte0='$', byte1='P', byte2='n', byte3='P'

Control Field - BIOSのイベント通知能力を示す.

bit[15-2]:	リザーブ(0固定)	
bit[1:0]:	イベント通知メカニズム	
	00	サポートされていない
	01	ポーリングによりサポート
	10	割り込みによりサポート

Event notification flag address - イベント通知メカニズムが上位ソフトウェアによるポーリングでサポートされているときのイベントフラグの物理アドレス.

Real Mode 16 bit offset to entry point -

Real Mode 16 bit code segment address -

16 bit Protected Mode offset to entry point -

16 bit Protected Mode code segment base address -

プラグ&プレイランタイムサービスルーチンのエントリアドレス

RAMとして持つ場合はサポートしないので参照しないこと.

OEM Device Identifier - システムを特定するためのID "PC98" (仮)

Real Mode 16 bit data segment address -

16 bit Protected Mode data segment base address -

プラグ&プレイランタイムサービスルーチンを利用する際, 呼び出しもとはこの情報を参照してセグメントディスクリプタを作成する. RAMとして持つ場合はサポートしないので参照しないこと.

3.8　従来ボードコンフィグレーションファイル

■概要

　従来の98バス拡張ボードのコンフィグレーションファイルフォーマットを規定する．これはEISA(Extended Industry Standard Architecture)のConfiguration File Formatをもとにしており，ここには変更箇所のみを記載する．したがって下記に記述していない事項についてはEISAの仕様を参照すること．

■仕様

●スロットステートメント

　スロットステートメントの値:valueはC98とする．このステートメントはオプションである．

記述例

SLOT=C98

●リバースステートメント/バーチカルステートメント

　スイッチステートメントおよびジャンパステートメントで記述するリバースステートメントとバーチカルステートメントの用法は次のリストを参照すること．
　リストにおいて上方はスイッチステートメントでの用例を示し，下方はジャンパステートメントを示す．
　なお，数値の記述は10進数，16進数とも0～9で始まることとする．

記述例:PC-9801-55(SCSIボード)の場合

```
BOARD
        ID = "NEC0551"
        NAME = "NEC PC-9801-55 SCSIインターフェースボード"
        MFR = "NEC"
        CATEGORY = "MSD"
        SLOT = C98
        AMPERAGE = 800
        COMMENTS = "NEC PC-9801-55 SCSIインターフェースボード"


SWITCH(1) = 8
        NAME = "SW1"
        STYPE = DIP
        VERTICAL = NO
        REVERSE = YES
        LABEL = LOC(1-8)      "1-ID0"
                              "2-ID1"
                              "3-ID2"
                              "4-ILV0"
                              "5-ILV1"
                              "6-ILV2"
                              "7-DMA0"
                              "8-DMA1"
        INITVAL = LOC(1-8) 111xxxxx
        FACTORY = LOC(1-8) 11111000

SWITCH(2) = 8
        NAME = "SW2"
        STYPE = DIP
        VERTICAL = NO
        REVERSE = YES
```

```
        LABEL = LOC(1-8)       "1-WND0"
                               "2-WND1"
                               "3-WND2"
                               "4-WND3"
                               "5-WND4"
                               "6-HST0"
                               "7-HST1"
                               "8-MSR"
        FACTORY = LOC(1-8) 01101101

JUMPER(1) = 1
        NAME = "SW3"
        JTYPE = TRIPOLE
        VERTICAL = YES
        FACTORY = LOC(1) 0

JUMPER(2) = 1
        NAME = "SW4"
        JTYPE = TRIPOLE
        VERTICAL = YES
        FACTORY = LOC(1) 0

FUNCTION = "SCSIインターフェースボード"

  SUBFUNCTION = "メモリアドレス"
       CHOICE = "DC000H-DDFFFH"
             FREE
             MEMORY = 8K
             ADDRESS = 0DC000H
             WRITABLE = NO
             MEMTYPE = OTH
             SIZE = WORD
             DECODE = 24
             SHARE = NO
```

```
        INIT = SWITCH(2) LOC(1 2 3) 011
CHOICE = "D4000H-D5FFFH"
        FREE
        MEMORY = 8K
        ADDRESS = 0D4000H
        WRITABLE = NO
        MEMTYPE = OTH
        SIZE = WORD
        DECODE = 24
        SHARE = NO
        INIT = SWITCH(2) LOC(1 2 3) 010
CHOICE = "EC000H-EDFFFH"
        FREE
        MEMORY = 8K
        ADDRESS = 0EC000H
        WRITABLE = NO
        MEMTYPE = OTH
        SIZE = WORD
        DECODE = 24
        SHARE = NO
        INIT = SWITCH(2) LOC(4 5) 01
CHOICE = "EE000H-EFFFFH"
        FREE
        MEMORY = 8K
        ADDRESS = 0EE000H
        WRITABLE = NO
        MEMTYPE = OTH
        SIZE = WORD
        DECODE = 24
        SHARE = NO
        INIT = SWITCH(2) LOC(4 5) 11
CHOICE = "ROM切り離し"
        DISABLE = YES
        FREE
```

```
            INIT = SWITCH(2) LOC(8) 0
SUBFUNCTION = "I/Oポートアドレス"
      CHOICE = "0CC0H-0CC4H"
            FREE
            PORT = 0CC0H
            PORT = 0CC2H
            PORT = 0CC4H
            INIT = JUMPER(1) LOC(1) 0
            INIT = JUMPER(2) LOC(1) 0
      CHOICE = "0CD0H-0CD4H"
            FREE
            PORT = 0CD0H
            PORT = 0CD2H
            PORT = 0CD4H
            INIT = JUMPER(1) LOC(1) 1
            INIT = JUMPER(2) LOC(1) 0
      CHOICE = "0CE0H-0CE4H"
            FREE
            PORT = 0CE0H
            PORT = 0CE2H
            PORT = 0CE4H
            INIT = JUMPER(1) LOC(1) 0
            INIT = JUMPER(2) LOC(1) 1
      CHOICE = "0CF0H-0CF4H"
            FREE
            PORT = 0CF0H
            PORT = 0CF2H
            PORT = 0CF4H
            INIT = JUMPER(1) LOC(1) 1
            INIT = JUMPER(2) LOC(1) 1

SUBFUNCTION = "DMAチャネル"
      CHOICE = "0"
            FREE
```

```
            DMA = 0
            SHARE = NO
            INIT = SWITCH(1) LOC(7 8) 00
      CHOICE = "2"
            FREE
            DMA = 2
            SHARE = NO
            INIT = SWITCH(1) LOC(7 8) 01
      CHOICE = "3"
            FREE
            DMA = 3
            SHARE = NO
            INIT = SWITCH(1) LOC(7 8) 11

SUBFUNCTION = "割り込みレベル"
      CHOICE = "9"
            FREE
            IRQ = 9
            SHARE = NO
            INIT = SWITCH(1) LOC(4 5 6) 110
      CHOICE = "3"
            FREE
            IRQ = 3
            SHARE = NO
            INIT = SWITCH(1) LOC(4 5 6) 000
      CHOICE = "5"
            FREE
            IRQ = 5
            SHARE = NO
            INIT = SWITCH(1) LOC(4 5 6) 100
      CHOICE = "6"
            FREE
            IRQ = 6
            SHARE = NO
```

```
            INIT = SWITCH(1) LOC(4 5 6) 010
      CHOICE = "12"
            FREE
            IRQ = 12
            SHARE = NO
            INIT = SWITCH(1) LOC(4 5 6) 001
      CHOICE = "13"
            FREE
            IRQ = 13
            SHARE = NO
            INIT = SWITCH(1) LOC(4 5 6) 101

  SUBFUNCTION = "本体タイプ"
      CHOICE = "8086/V30機仕様"
            FREE
            INIT = SWITCH(2) LOC(6 7) 10
      CHOICE = "PC-98XA"
            FREE
            INIT = SWITCH(2) LOC(6 7) 00
      CHOICE = "N/H共用機仕様"
            FREE
            INIT = SWITCH(2) LOC(6 7) 01
      CHOICE = "80286/386機仕様"
            FREE
            INIT = SWITCH(2) LOC(6 7) 11
```

3.9　従来互換モード

プラグ&プレイ非対応の従来のハードウェアで使用する場合やデバイスドライバなどを介さず，直接I/Oを設定されることが多い種類のボードでは，従来ボード互

換のモードを設ける必要がある．これらについては下記の方法を推奨する．

・プラグ&プレイモードと従来互換モードを持ち，これらのモードはスイッチで切り換える．このスイッチは本体に設置していても操作できるようにしておく．
・プラグ&プレイモードでは本来のプラグ&プレイボードとして動作する．
・従来互換モードでは，従来ボードと同じリソースを使用する．設定はボード上に搭載するE2PROM等の不揮発メモリに記録しておき，パワーオン/リセット時にボード内でリードして自己設定する．
・不揮発メモリへのアクセスはプラグ&プレイモードで行なう．使用可能リソースは"Vendor Defined"を使用して拡張したプラグ&プレイリソースデータ内に記述する．アクセスは下記フローのように行なう．
・ボード固有の設定に関しては，さらに拡張して不揮発メモリに設定するか，その部分のみDIP SW等で行なう．

不揮発メモリへの設定フロー

Plug and Playモードでパワーオン

↓

アイソレート/リソースデータ読み込み

↓

ユーティリティで設定

↓

互換モードにしてリセット

↓

従来ボードとして動作

補足

1. プラグ&プレイ対応機種

PC-9821An以降の機種はプラグ＆プレイとなっており，プラグ＆プレイに対応した拡張ボード，カード等を利用することができる.

以下の機種のWindowsインストールモデルでは，プラグ＆プレイを実現するためのソフトウェア/データファイルがインストール済みである(フレームモデルはオプション).

98MATE
PC-9821Xt13
PC-9821Xa12
PC-9821Xa10
PC-9821Xa9
PC-9821Xa7
PC-9821Xe10
PC-9821Xf
PC-9821Xa
PC-9821Xt
PC-9821Xn
PC-9821Xp
PC-9821Xs
PC-9821Xe
PC-9821Ap3
PC-9821As3

98NOTE
PC-9821Ne3
PC-9821Nd2
PC-9821Na7
PC-9821Nx
PC-9821Np
PC-9821Ns
PC-9821Ne2
PC-9821Nd
PC-9821Nf
PC-9821Nm
PC-9821NL/A

98NOTE LIGHT
PC-9821Lt2
PC-9821Lt
PC-9821Ld

98FELLOW
PC-9801BA3
PC-9801BX3

98FiNE
PC-9821Es

●動作環境

- MS-DOS Ver5.0A／6.2
- Microsoft Windows 3.1
- Microsoft Windows 95(対応予定)

1.1　拡張ボードにおける
プラグ&プレイのメリット

提供されるプラグ&プレイサポートソフトウェアおよび，プラグ&プレイ対応拡張ボードを利用することにより，次のようなメリットが得られる.

・プラグ&プレイ対応オプションボードにおける動作環境設定が不要になる.
→プラグ&プレイ対応オプションボードにおけるスイッチ設定操作(割り込みチャネル，IOポートアドレス，拡張ROMアドレス)が不要

注意：従来のボードは，従来通りユーザがボード上のディップスイッチ/ジャンパスイッチを設定

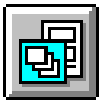
・システム資源(割り込みチャネル，IOポートアドレス，拡張ROMアドレス等)をソフトウェアにより管理可能となる.
→使用しているシステムの資源を画面上で確認できる.
(従来はユーザがシステム資源の管理をしなければならない.
サードパーティー製の従来ボードは画面の指示に従い入力する)

1.2　プラグ&プレイ関連ソフトウェア

　Windowsインストールモデルでは，プラグ&プレイを実現するためのソフトウェア/データファイルがインストール済みである.

・コンフィグレーションマネージャ(CM)
・コンフィグレーションユーティリティ(CU)
・リソースデータファイル

▼*プラグ&プレイ関連ソフトウェア*



■コンフィグレーションマネージャ

　コンフィグレーションマネージャ(CM)は，プラグ&プレイに対応した汎用拡張スロット(Cバス)用ボードのH/Wリソース(割り込みチャネル，IOポートアドレス，拡張ROMアドレス等)の自動割り当てを実現する.

注意：・自動設定可能なボードはプラグ&プレイ対応ボードのみ.
　　　・プラグ&プレイ未対応のボードを使用する場合には，コンフィグレーションユーティリティを使用し，設定値を指定した後に，従来同様ボード上のスイッチ等で設定を行う必要がある.
　　　・本ソフトは各ボードのハードウェア設定を行うためのものであり，各ボードを制御するソフトウェアドライバの自動組み込み等はおこなわない.

■コンフィグレーションユーティリティ

コンフィグレーションユーティリティ(CU)により，設定値の確認変更が可能となる.

・自動割り当てされた設定値の確認，変更が可能.
　現在の設定値の確認，プラグ&プレイ対応ボードの設定変更が可能.
・プラグ&プレイ未対応ボードの設定すべき値の確認，マニュアルでの設定可能な値の確認が可能.
・プラグ&プレイ対応ボード(PCIボード，プラグ&プレイ対応Cバス用ボード，PCカード)と未対応ボードの混在使用時のH/Wリソースの重なりを避けることが可能.
・MS-DOS用とWindows用が用意されている.

■リソースデータファイル

リソースデータファイルは，既存ボード(プラグ&プレイ未対応)のセットアップデータ(割り込みチャネル，IOポートアドレス，拡張ROMアドレス等)を記述したファイルである.

・プラグ&プレイサポートソフトウェアを使用している環境で，プラグ&プレイ未対応の従来オプションボードをセットアップする場合に本ファイルが参照され，自動的にH/Wリソースのぶつかり合いのチェックが行われ，設定可能な資源が決定される.
→ユーザは，決定された設定値に従いボード上のスイッチ等を設定するだけでよく，本体や他のボードとのぶつかり合いがないかを考えなくても良い.

注意：各ボード上で資源(割り込みチャネル，IOポートアドレス，拡張ROMアドレス等)の衝突が発生しないように管理することが目的であり，ボード上のスイッチの設定は従来通り必要である.

■PCカードサポートユーティリティ

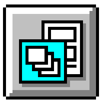
PCカードサポートソフトウェアはJEIDA Ver4.2,PCMCIA2.1に対応したカードを利用するためのドライバおよびユーティリティである.

JEIDA Ver4.1,PCMCIA2.0用のドライバ,アプリケーションに対しての互換性を確保するドライバも用意されている.

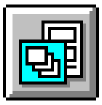
詳細は「アプリケーション開発編 第4部 PCカード」を参照のこと.

1.3　拡張ボード設定フロー

▼*拡張ボード設定フロー(プラグ&プレイ対応)*



▼*拡張ボード設定フロー(プラグ&プレイ未対応)*



設定フロー注釈

*1　プラグ&プレイに対応していないボードを使用する場合には，従来同様に手動で設定する手順は残る．

*2　現在のＯＳ環境(MS-DOS, Windows3.1)では，デバイスドライバの自動登録はできない．

2.Windows 95対応機種

■PC95対応機種

Windows 95対応(PC95の規格に適合)として発表された機種を以下に示す.

98MATE

PC-9821Xt13/C12
PC-9821Xa12/C8
PC-9821Xa12/C12
PC-9821Xa10/C4
PC-9821Xa10/C12
PC-9821Xa9/C4
PC-9821Xa9/C8
PC-9821Xa7/C4
PC-9821Xa7/C8
PC-9821Xa7e/S17
PC-9821Xa7e/S15
PC-9821Xe10/C4
PC-9821Xe10/S15

98MULTi

PC-9821Cx2/S15B
PC-9821Cx2/S15T
PC-9821Cx2/S17B
PC-9821Cx2/S17T
PC-9821Cb2/B
PC-9821Cb2/T
PC-9821Cb2/M
PC-9821Cb2/A

98NOTE

PC-9821Na7/H3
PC-9821Na7/HC7
PC-9821Nx/3
PC-9821Nx/C7
PC-9821Ne3
PC-9821Ne3/3
PC-9821Nd2
PC-9821Nd2/3

98NOTE LIGHT

PC-9821Lt2/3D
PC-9821Lt2/3A
PC-9821Lt2/7A

PC95の規格については本編「第2部 第1章 システムデバイス」,及び弊社発行の「Windows 95 ハードウェアデザインガイド」を参照.

■推奨仕様

PC-9821Xa10,Xa9,Xa7以降の機種は,Windows 95(PC95)推奨仕様に対応している.以下に対応箇所を示す.

・高速シリアルインターフェイス

115,200bpsの通信速度をサポート.2チャンネル装備する機種については,チャネル2に赤外線通信インターフェイスユニット(オプション)を接続することにより,SIR方式及びASK方式の赤外線通信(IrDA規格準拠)が可能.

・赤外線通信機能

IrDA規格準拠の赤外線通信機能.115,200bpsの通信速度をサポートする.

・DDC対応CRTインタフェース

▼*CRT I/F 端子番号と信号線(アナログRGB,ミニD-SUB15ピン)*

●対応機種

以下に対応機種の一覧を示す.

▼*対応機種一覧*

本編の第２部 第３章「PC98バスプラグ＆プレイ仕様」は“Plug and Play ISA Specification”を,インテルジャパン株式会社およびマイクロソフト株式会社の了解の基に日本電気株式会社がPC-９800シリーズ用に変更作成した資料から転載したものです.